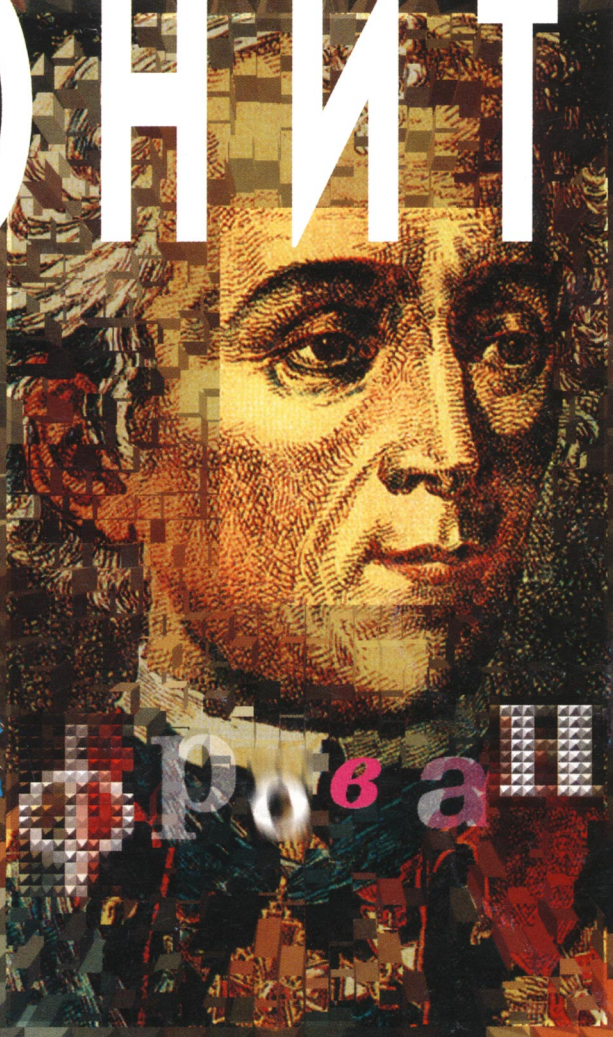
МОНИТОР

1.95

ISSN 0869-3889



Шрифтование

МОНИТОР
Аспект

Аппаратура

Нейрокомпьютеры

Оценка производительности вычислительных систем

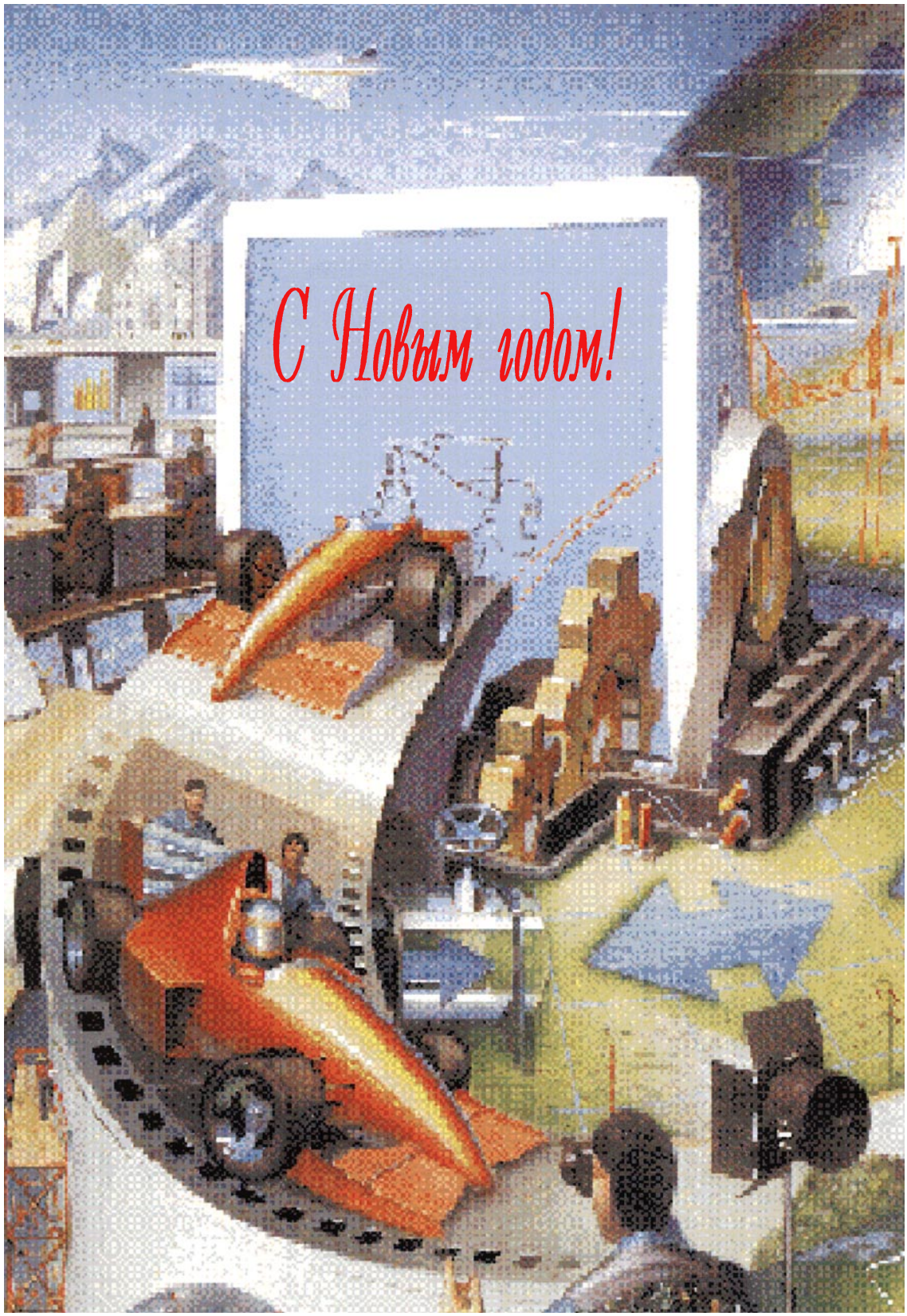
ПРОГРАММЫ

СУБД HyTech 2.0

ТЕХНОЛОГИИ

Распознавание образов: уточнение

Гибкая статистическая компоновка



Autodesk®

Постоянные читатели нашего журнала, вероятно, помнят, что в первых номерах «Монитора» (1991–1992 годы) статьи для программистов чередовались со статьями для пользователей («Сколько экспертных систем уместится на кончике иглы», «Куда катится компьютерное яблоко», «Портативные компьютеры», «Обзор графических редакторов IBM PC», «Новое лицо редактора ChiWriter — версия 4!») и со статьями на экономические и юридические темы («Компьютерный бизнес в СССР», «Юридическая защита программ», «Рынок программных продуктов в России: первые шаги», «Регистрация объектов авторского права в США»).

К концу 1992 года мы пришли к выводу, что журнал для программистов объемом всего 64 страницы, часть из которых занята рекламой, не может позволить себе публикацию статей, отклоняющихся по тематике от основного направления журнала. Исключение было сделано только для рубрики «Игры». Вместе с тем, учитывая, что к нам в редакцию стекается значительное число хороших «непрограммистских» статей на компьютерные темы, мы начали издание отдельного журнала «Монитор–Аспект» с периодичностью 4 номера в год, в котором и сосредоточились упомянутые статьи.

Два года параллельной работы над «Монитором» и «Монитор–Аспектом» привели нас к следующим выводам.

Минимальная нормальная периодичность для журнала, освещающего столь динамичную область, как компьютеры, — 8 номеров в год. При меньшем числе номеров статьи устаревают.

Издавать параллельно два журнала нормальной периодичности силами одной редакции чисто организационно достаточно сложно.

Унаследованный «Монитор–Аспектом» от «Монитора» серьезный подход к излагаемому предмету приводит к существенному пересечению читательской аудитории двух журналов: программисты обычно читают статьи, посвященные аппаратуре и компьютерным технологиям, для многих (научные работники, инженеры) программирование является инструментом для решения профессиональных задач, не решаемых стандартными программами, в остальном же они остаются пользователями, ряд тем крайне затруднительно объявить чисто «программистскими» или «пользовательскими», так как они интересны всем, например операционные системы или глобальные сети.

Иными словами, граница между программистом и квалифицированным пользователем то и дело нарушается с обеих сторон, а делить статьи по принципу «Все, что с листингами — в «Монитор», все, что без листингов — в «Аспект»», то есть считать, что программист не способен читать ничего, кроме листингов, а пользователь, напротив, не способен разобраться даже в тексте простейшей программы, — весьма далекое от реальной жизни упрощение.

Это подтверждается и проведенным в редакции анализом базы данных наших подписчиков: очень много читателей подписывается одновременно на оба журнала.

Взвесив все эти соображения и оценив имеющуюся у нас сегодня возможность увеличить объем и периодичность журнала «Монитор», мы приняли решение увеличить число выпусков «Монитора» до 10 в год, объем журнала — до 128 страниц (как минимум), а «Монитор–Аспект» сделать «журналом в журнале» — включить его отдельным блоком в состав журнала «Монитор». Основная направленность «Монитора» как журнала для программистов–практиков остается неизменной, а квалифицированные пользователи, как мы надеемся, найдут полезными и интересными статьи в «Монитор–Аспекте», тем более что 10 выпусков «Аспекта» позволят публиковать статьи гораздо оперативнее.

Мы понимаем, что часть наших читателей, относящихся к «химически чистым» программистам или пользователям, такое решение не вполне устраивает, однако в силу изложенных выше обстоятельств нам оно представляется оптимальным.

Те, кто подписался на 8 номеров «Монитора» на 1995 год, могут дополнительно подписаться на номера 9 и 10 там же, где они оформляли подписку на номера 1 – 8.

Мы приносим извинения нашим подписчикам за неудобства, связанные с изменением объявленных ранее условий подписки на 1995 год, и обещаем, что наш отдел распространения совместно с организациями, проводящими подписку, сделает все возможное для устранения возникших недоразумений.

Д. Э. Бройтман, главный редактор

Реклама в номере

Apple Computer	97	Nexus	110, 111
ARUS	109	Software Security, Inc.	44
Autodesk RF	2–я обл.	Trans–Ameritech Enterprises, Inc.	33
CROCUS Int.	92, 93	VIMCOM	3–я обл.
CROC, Inc.	45	WP International, Inc.	41
DEC	89	YAM Int.	121
ELVIS PLUS	21	АО МКЦ "Вариант"	7
Fitec	44	ИНТЕРЛАЗ	98
LVS Corporation	15, 100	МЦНТИ	125
Merisel	4–я обл.	СКТБ Компьютерных Сетей	101
NATEKS	78, 125	СП.АРМ	125

Ответственность за информацию, приведенную в рекламных материалах, несет рекламодатель

Наш компьютерный ларек вы найдете на стр. 102–103.

Журнал для программистов–практиков

Редколлегия:

Д.Э. Бройтман
(главный редактор)
(brtmn@mntr.msk.su)
В.В. Водолазкий
(vodolaz@mntr.msk.su)
В.С. Головин
Г.Л. Карчикян
(giga@mntr.msk.su)
Н.Н. Косарев
А.М. Миляков
(ответственный секретарь)
А.Р. Мурадян
(зав. тестовой лабораторией)
О.А. Пригоникер
Е.В. Равич
Л.И. Турок (turok@mntr.msk.su)

Отдел рекламы:

Н.А. Никитина
(заведующая отделом)
В.Ю. Томская–Ефремова

Графическое бюро:

С.В. Макельский
(главный художник)
Д.В. Герасимова
Ю.А. Исаева
А.В. Семенников
(компьютерная верстка)

Компьютерный «ларек»:

А.З. Морев (morev@mntr.msk.su)

Отдел новостей:

А.В. Лепехин

Отдел распространения:

М.Е. Желаннова
(заведующая отделом)
Р.А. Болдырев
А.А. Мишустин
М.А. Сычев

Дирекция:

А.В. Орехов
(издательский директор)
А.Д. Гассельблат
(коммерческий директор)

В оформлении номера использованы рисунки
А.С. Замулы

Для писем: 117419 Москва, а/я 765
E-mail: post@mntr.msk.su
Редакция: (095) 238–99–33
Факс: (095) 238–99–33
Отдел рекламы: (095) 903–46–26

Учредитель:
АО "Монитор"

Журнал зарегистрирован Комитетом Российской Федерации по печати 20.07.94. Регистрационный номер 012752

Подписано в печать с оригинал–макета 28.12.94
Формат 60x84/8.
Отпечатано в типографии «Oy Scanweb Ab» (Финляндия)

© «Монитор», 1995

Содержание

Письма		Задачник «Монитора»	5
		О рациональном использовании DoubleSpace	
Технологии программирования	С. Короткий.	Введение в распознавание образов Часть 2. Утончение	8
	С. Шмаков.	Шаблон программ обработки файлов	12
	С. Короткий.	Гибкая статическая компоновка	16
Базы данных	А. Дубина, С. Сынгур.	Событийно-управляемый интерфейс в СУБД FoxPro	18
	С. Ковалевский, А. Малярский, К. Зайцев, С. Ушенин, В. Нечаев, В. Хруст.	СУБД HyTech 2.0: архитектура и принципы работы	22
Языки программирования	К. Шереметьев	Прыжок Ягуара	31
MS BASIC	А. Колесов.	Смешанное программирование и подключение универсальных библиотек в Basic программах	34
DOS под микроскопом	В. Рабец.	Если в CMOS испортились параметры винчестера	42
Windows	В. Плотников.	Создание программной оболочки в среде Windows	46
UNIX	В. Водолазкий.	Как без головной боли и нервотрепки установить Linux Часть 2	53
	Н. Кожекин.	Новое поколение выбирает UNIX!	57
Защита	А. Винокуров.	ГОСТ не прост..., а очень прост!	60
	Д. Герасимчук.	О пользе «ассемблерных извращений»	74
Коммуникации	Д. Мирошников, С. Воеводский, Д. Воеводский.	Что такое «модем для России»?	76

Содержание

Рабочие станции

Д. Волков.
Методы оценки производительности вычислительных систем 80

Корпорации

Партнер, которому отдают предпочтение 90

Специализированные компьютеры

А. Масалович.
Нейронные сети на пути к российскому рынку 94

Персональные компьютеры
104
112
119
122

Рабочие станции 80

Сети

А. Замятин.
Применение программного обеспечения удаленного доступа
The Norton pcANYWHERE 104

В статье рассматривается семейство программных продуктов The Norton pcANYWHERE 4.5 (Host, Remote и LAN). Подробно описано их конфигурирование для случая прямого соединения через последовательный асинхронный интерфейс.

Мультимедиа

В. Дьяконов.
Звуки мультимедиа в среде MS-DOS 112

Об установке драйверов звуковой платы мультимедиа в среде MS-DOS, тестировании и работе.

Д. Мокин.
Живое видео на экране компьютера 119

Игры

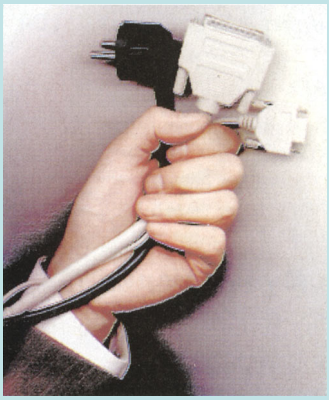
О. Стариков.
«Hero Quest III» — путь к славе 122

Новости

126

Господа производители аппаратного и программного обеспечения!

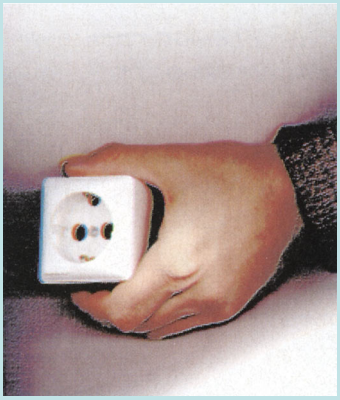
это Вы



это Мы

Тестовая лаборатория

это Они



а это результат Нашего сотрудничества

Представляем Вам Независимую Тестовую Лабораторию (НТЛ).

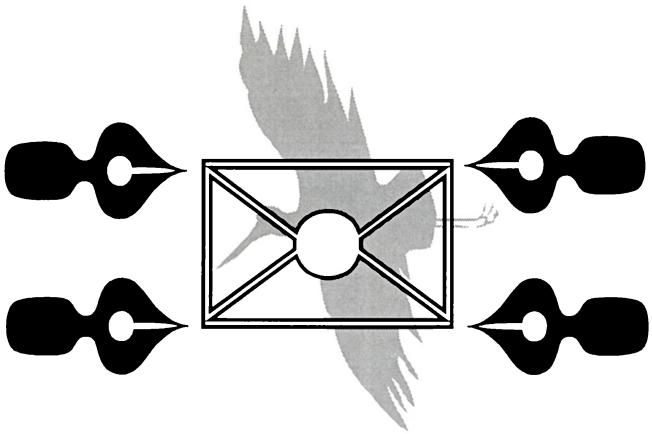
НТЛ проводит тестирование и анализ программных продуктов, компьютеров, аппаратуры для компьютеров, периферийных устройств, сетей, вычислительных комплексов.

НТЛ публикует результаты таких испытаний в печати.

НТЛ вырабатывает свод критериев качества **НТЛ** и выдает разработчикам, производителям и другим заинтересованным лицам и организациям сертификаты о соответствии их продукции своду критериев качества **НТЛ**.

Контактные телефоны НТЛ: 238-99-11 («Монитор»); 284-57-28 («АС»).

МонитОр



Задачник «Монитора»

В 7-м номере «Монитора» за 1994 год были подведены итоги второй части конкурса, в которой определялась самая короткая подпрограмма, выводящая на экран число в шестнадцатеричном виде. Победителем данного вида многоборья стал Вячеслав Львович Семенов из г.Череповца, предложивший решение объемом 21 байт. Присланное им решение казалось настолько компактным, что даже «старорежимные» программисты отказались от попыток улучшить его. И, как выяснилось, напрасно. В редакцию было прислано решение, превосходящее чемпионский результат на 2 байта, а автором этого решения стал (угадайте с трех раз) **Вячеслав Львович Семенов!**

Хотя конкус официально завершен, ознакомиться с этим решением просто необходимо всей программирующей общественности. Чтобы оценить всю его красоту, давайте рассмотрим задачу с самого начала.

Итак, нужно написать подпрограмму, которая выводит на экран число в шестнадцатеричном виде, которое передается в регистре AX. Очевидно, что все преобразования числа необходимо проводить в этом же регистре — это не судьба, просто так организована система команд процессоров серии 80x86 и система прерываний DOS (короче int 29h не придумано еще ничего). Все преобразования выполняются в два шага. На первом нибл (группа из четырех битов) помещается в младшие разряды регистра AX, а на втором этот нибл преобразуется, в зависимости от своего значения, либо в цифру, либо в букву. Ну, плюс к тому, еще надо организовать цикл для последовательного перебора всех четырех ниблов, начиная с самого старшего.

Как справедливо отметил В. Семенов, «организация цикла Ю. Пером выглядит предпочтительнее», потому что у него наиболее компактно взаимоувязаны две задачи — собственно организация цикла и выделение соответствующего нибла.

```
Show_hex  proc near
          mov  cl,12
N_c:      push ax
          shr  ax,cl
          ...
          int  29h
          pop  ax
          sub  cl,4
          jae  N_c
          retn
Show_hex  endp
```

По-моему, данный фрагмент программы настолько очевиден, что не нуждается в комментариях и полностью совпадает с решеним Ю. Пера («Монитор» 7'94).

Итак, осталась только одна задача — преобразовать значение младшего нибла либо в цифру, либо в букву. Напомню, что в предыдущей версии данной подпрограммы такое преобразование у В. Семенова выглядело следующим образом:

```
          aaa
          jae  N_v
          add  al,11h
N_v:      add  al,30h
```

В этом маленьком фрагменте сразу бросаются в глаза две команды сложения, которые выглядят несколько тяжеловесно, и, скорее всего, сэкономить можно именно на их оптимальной организации. Дальше для пояснения решения воспользуюсь письмом В.Семенова.

«В основу положено решение системы уравнений:

```
X    *Y = 48 mod 256
(X+1)*Y = 65 mod 256
```

Минимальные значения X и Y, при которых система имеет решение, таковы: X=48, Y=17.»

Маленькое замечание: 48 — это код символа нуля, а 65 — код символа буквы A.

«Имея эти исходные данные и описание команд процессора AAA и AAD, достаточно просто понять процесс преобразования 4 младших битов регистра AX в символьный вид.

Команда

```
mov  ah,48h
```

подготавливает значение старшего байта для команды символьной коррекции AAD (то есть выполняется присваивание X=48 — Д.П.).

Команда

```
aaa
```

модифицирует регистр AL, как описано в «Мониторе» 7'94, но одновременно, если значение 4 младших битов AL было больше 9, то значение AH увеличивается на 1.»

После выполнения этих двух команд
AH=48 и 0=<AL=<9, если значение младшего нибла было меньше 9;
AH=49 и 0=<AL=<6, если значение младшего нибла было больше 9.

«Команда

```
aad (db 0D5h,17)
```

выполняет вычисление по формуле AL=AH*Y+AL, что позволяет сократить команду дополнительного сложения.»

В результате подпрограмма имеет следующий вид:

```
Show_hex  proc near
     mov  cl,12
N_c: push ax
     shr  ax,cl
     mov  ah,48h
     aaa
     db   0D5h,17
     int  29h
     pop  ax
     sub  cl,4
     jae  N_c
     retn
Show_hex  endp
__________ Итого 19 байтов.
```

Остается заметить единственное, что «может собственных Платонов и быстрых разумом Невтонов российская земля рождать».

Д. Правиков

■

Уважаемая редакция.

Предлагаю способ рационального использования DoubleSpace. Везде пишут, что DoubleSpace открывает второе дыхание компьютерам «стандартной» конфигурации — «286\1\40...». Но при установке его на такой компьютер увеличивается дисковая память, свободное ОЗУ уменьшается. Было бы легче, если бы его можно было отключать. Конечно, можно нажимать Ctrl-F8 и удалять dblspace.sys. Но если необходимо убирать на некоторый срок, то мой способ более приемлем. Суть его в том, что если удалить dblspace.ini, то dblspace.bin грузиться не будет.

У себя я создал меню в config.sys, где мне предоставляется выбор: «начать сезон DoubleSpace» или «закончить его».

Сразу хочу пояснить, что я создал диск E, сжав свободное место на диске C. На этот диск я записываю хорошо поддающуюся компрессии информацию: текстовые файлы и программы вроде PROMT (около 7 мегабайтов развернутый вариант и 900 килобайтов сжатый).

Чтобы все работало нормально, необходимо создать копию dblspace.ini — dbl_ini.my, но при этом убрать атрибуты hidden, read-only и system. Вот все мои файлы конфигурации:

config.dbl

```
[Menu]
menucolor=15,0
menuitem=dbl,Start DublSpace season
menuitem=pr,*Ваша конфигурация*
menudefault=pr,2

[pr]
*  Ваши драйверы*
[dbl]
```

config.nds

```
[Menu]
menucolor=15,0
menuitem=ndbl,End DublSpace season
menuitem=pr,*Ваша конфигурация*
menudefault=pr,2
[common]
device=c:\dos\dblspace.sys /move

[pr]
* Ваши драйверы*
[ndbl]
```

autoexec.bat

```
p$g
PATH C:\DOS;C:\WINDOWS;D:\pkware;e:\tasm;e:\tasm\work
SET TEMP=C:\TEMP
goto %config%

:dbl
copy config.nds config.sys
copy dbl_ini.my dblspace.ini
attrib +r +h +s dblspace.ini
reb

:ndbl
copy config.dbl config.sys
attrib -r -h -s dblspace.ini
del dblspace.ini
reb

:pr
* Ваши драйверы*
```

При установке этой системы необходимо скопировать config.nds в config.sys.

Еще вам понадобится программа reb. Её можно создать с помощью ассемблера (int 19h) или перезагружать компьютер вручную (Ctrl-Alt-Del). ■

Росляков Андрей,
Москва, лицей 1502

БЫСТРО, ПРОСТО, НАДЕЖНО

"ZyXEL многих заставил думать по-другому"
- из переписки в конференции RELKOM. POSTMASTERS

PC MAGAZINE EDITORS' CHOICE
June 29, 1993
ZyXEL U-1496 PLUS

JULY 1993 BYTE BEST VALUE HIGH-SPEED MODEM
ZyXEL U-1496E PLUS

Data Comm MAGAZINE Top Performer JULY 1993
ZyXEL U-1496 PLUS

COMPUTER SHOPPER
July 1993, ZyXEL U-1496E PLUS
ZyXEL U-1496 PLUS
Top 10 List

JULY 1993 BYTE BEST TWO-WAY ALL-AROUND MODEM
ZyXEL U-1496E

PC PROFESSIONELL EMPFEHLUNG DER REDAKTION
February 1993
ZyXEL U-1496E PLUS

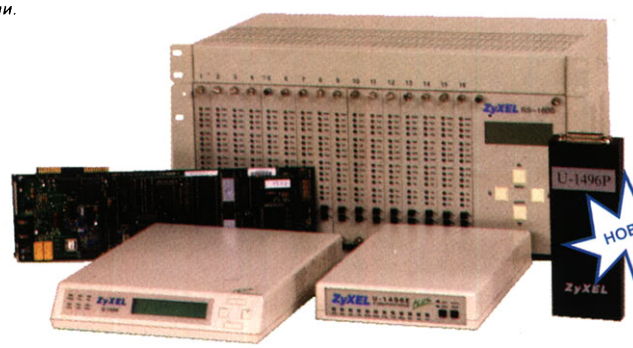
Некоторые считают, что ZyXEL - самый хороший из быстрых модемов. Другие считают, что ZyXEL - самый быстрый из хороших модемов. Мы согласны и с тем, и с другим.

Вы спросите почему? Причины очевидны: Главный критерий оценки модема - надежность его работы. Именно поэтому ZyXEL стал стандартом de-facto для высокоскоростных модемов в России. Банки, торговые дома, транспортные компании, предприятия топливно-энергетического комплекса, сети передачи данных - все, кто знает цену своей информации, доверяют ее модемам ZyXEL.

19200 для модемов ZyXEL - это не просто внушительное число, написанное на коробке, а реальная физическая скорость, которая дает вам экономию времени и денег.

Более того, удивительно широкий набор дополнительных функций модема позволит вам сразу решить и другие коммуникационные проблемы: прием и рассылка факсов, организация речевой почты (interactive voice mail) и комплексных информационных систем.

Широкий спектр моделей серии U-1496 полностью соответствует многообразию задач, которые стоят перед вами. Семейство модемов в промышленном исполнении и NMS позволят осуществить контроль и управление сетью из сотен модемов одновременно из одного места. Для тех, кто не может позволить себе быть без связи в дороге, новый портативный модем U-1496P станет незаменимым дополнением к переносному компьютеру и сотовому телефону.



Все модемы ZyXEL совместимы с большинством других модемов и могут работать в синхронном или асинхронном режиме с такими программными средами, как DOS™, Windows®, OS/2®, Macintosh®, UNIX®, NeXT®, Amiga™, Atari™. В синхронном режиме с использованием команд V.25bis. ZyXEL совместим с системами AS/400® и RS/6000® фирмы IBM.

Основные характеристики модемов ZyXEL:

- *Высокая скорость 19200 bps*
- *Сверхнадежный режим ZyCELL*
- *V.32bis/V.32, V22bis/V.22, BELL 212A*
- *V.17 - 14400 bps, ELA Class 2, G3 Fax, S/R*
- *V.42/V42.bis (+ Selective Reject), MNP® 3/4/5*
- *Цифровая запись/воспроизведение звука*
- *АОН, распознавание условных звонков*
- *Дистанционное конфигурирование*
- *Динамический выбор рабочей скорости*
- *Защита от НСД: пароли, обратный звонок*
- *Адаптация к абонентской линии*
- *2/4 проводная коммутируемая/выделенная линия*
- *Автоматическое распознавание режима Data/Fax/Voice*
- *Перепрограммируемое ПЗУ —обновление микропрограммы*

yes
It runs with NetWare

Сертификат Министерства Связи России

Приобретая модемы ZyXEL у авторизованных дилеров МКЦ "Вариант" вы, кроме всего этого, получите оборудование, произведенное специально для России, имеющее сертификат Министерства Связи, необходимые дополнительные программы, 2 года гарантии со склада, бесплатное обновление микропрограммы, профессиональную поддержку квалифицированного технического персонала и доступ к BBS.

ZyXEL

Официальный дистрибьютор
АО МКЦ "Вариант"
117279, Москва, Островитянова 37а
Тел. (095) 420 2519
Факс.(095) 420 5311

Круглосуточно
Информация (095) 932 8510
Электронная почта
max@mechta.msk.su, 2:5020/22@fidonet
WHITE BEAR BBS - (095) 932 8465

Введение в распознавание образов

С. Короткий

Часть 2. Утончение

На протяжении тысячелетий задача распознавания образов успешно решается зрительной системой человека, оставаясь для самого человека, по сути, тайной за семью печатями. Точность и быстрота человеческого восприятия объясняются, по-видимому, не только специализацией различных областей зрительной коры головного мозга, но и способностью мозга обобщать увиденное, моментально переводя его в абстрактные понятия [1]. Это замечательное свойство должно быть в той или иной степени смоделировано на компьютере для успешной автоматизации процесса распознавания. Сложность данной задачи не позволяет решить ее сразу в общем виде и вынуждает ограничиться рассмотрением планарных линейчатых изображений, к которым могут быть преобразованы практически все более сложные, даже трехмерные изображения.

После ввода графической информации в компьютер большинство линий, составляющих изображение, приобретает размытый вид. Например, на рисунке 1 представлен квадрат с вписанной в него окружностью, полученный путем сканирования чертежа. Такое качество изображения не позволяет применять к нему известные [2] методы по выделению тех или иных графических примитивов, в то время как глаз легко адаптируется к чрезмерной толщине и неровности линий. В связи с этим для выделения остовов кривых, несущих основную информацию об их форме и взаимосвязях, планарные изображения обычно подвергают утончению. Фактически оно эквивалентно удалению излишней информации без потери сущности и связности элементов изображения. Таким образом, утончение можно отнести к первому этапу обобщения информации в алгоритме распознавания изображений. На следующих этапах с помощью преобразования Хафа и других методов можно достичь степени абстрагирования, достаточной для введения понятий графических примитивов с соответствующими параметрами, набор которых, по сути, и является выходными данными процедуры распознавания.

Существует большое число алгоритмов, проводящих утончение или прореживание. Их можно классифицировать по порядку обследования пространства на последовательные и параллельные, по точности выделения остовов — на точные и приближенные, а также по методам выбора остовных точек. Последовательные и параллельные

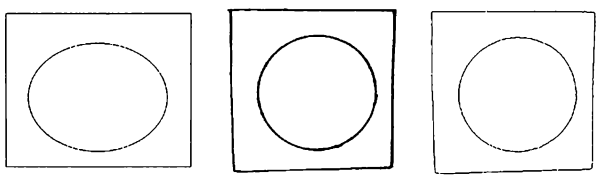


Рис. 1. Изображение идеального качества и его представление после сканирования и утончения

алгоритмы относятся к разряду итерационных, то есть выполняющих циклы, при осуществлении каждого из которых у всех фигур удаляются только точки наружного слоя. В последовательном алгоритме точки исследуются на предмет их удаления в твердо установленном порядке, так что удаление точки на n-й итерации зависит от результатов (n-1)-й итерации и состояния точек, уже обработанных на n-й итерации. В параллельном алгоритме удаление зависит только от состояния изображения после (n-1)-й итерации и все точки могут исследоваться независимо в произвольном порядке.

Многие классические алгоритмы утончения используют так называемые конфигурации — структуры ближайшей окрестности рассматриваемого пиксела, совпадение которой с расположением точек на изображении приводит к пометке данного пиксела как оставляемого, а несовпадение — к его пометке для удаления. В [3] описан один из таких алгоритмов, являющийся последовательным, на основании которого была написана программа, приведенная в листинге. На рисунке 2 показаны используемые в алгоритме конфигурации окрестностей центрального пиксела. Хотя бы один из пикселов каждой группы, помеченной символом A или B, должен быть отличным от нуля, а пиксел, помеченный цифрой 2, уже оставлен как принадлежащий к остову. Каждая из двух показанных конфигураций инвариантна относительно ориентации, то есть проверки необходимо проводить с последовательными поворотами конфигураций на 90 градусов: первую — для двух ориентаций, вторую — для четырех. Нумерация точек внутри конфигураций производится в соответствии с рисунком 3.

Для дальнейшего изложения необходимо ввести несколько определений. Все точки, входящие в конфигурацию, кроме центральной, называются 8-соседями. Точки 0, 2, 4, 6 называются 4-соседями. Контурные или граничные точки, исследуемые на предмет удаления, в большинстве методов определяются как имеющие по крайней мере одного белого 4-соседа. Неконтурные точки называются внутренними. Теперь мы можем дать новое определение итерации. Итерацией называется один цикл анализа изображения с удалением всех текущих контурных точек. Общее число черных точек в окружении точки p обозначается через B(p), а среди 4-соседей — B'(p). Очевидно, что конечная точка любой линии будет иметь B(p)=1. Далее нам потребуются так называемые параметры пересеченности.

$$R(p) = \sum_{i=0}^{7} | x((i+1) \bmod 8) - x(i) | = 2B'(p)$$

$$H(p) = \sum_{i=0}^{3} b(i), \text{ где}$$

$$b(i) = \begin{cases} 1, \text{ если } x(2i) = 0 \text{ и } (x(2i+1) = 1 \text{ или } x(2i+2) = 1) \\ 0 \end{cases}$$

Здесь x(j) — значение пиксела j, то есть 1, если он черный, и 0, если он белый.

Точка p удаляема, если: 1) p — черная, 2) B(p)>1, 3) p — контурная. В ходе одной итерации точки, удовлетворяющие этим условиям, не удаляются, а помечаются. Удаление производится только после анализа всех точек.

Алгоритм Хилдича [4], просматривающий изображение слева направо и сверху вниз, использует 4 дополнительных условия для отметки точки:

1) по крайней мере один черный сосед не должен быть помечен;
2) в начале итерации H(p)=1;
3) если точка 2 помечена, установка x(2)=0 не должна менять H(p);
4) то же самое для точки 4.

Алгоритм Рутовица [4] применяет следующие условия:

1) если R(p)=0 или 8, p не удаляется;
2) если R(p)=2 и x(0)×x(2)×x(4)×x(6)=0 и B(p)>1, p удаляема;
3) если R(p)=4 и x(0)×x(2)×x(4)×x(6)=0 и $\sum_{i=0}^{3} x(2i) \times not(x(2i+1)) \times x(2i+2) = 1$, p удаляема;
4) если R(p)=6 и $\sum_{i=0}^{3} x(2i) = 3$, p удаляема.

Перейдем к рассмотрению параллельных алгоритмов. Простые параллельные алгоритмы утончения с окном соседей размером 3 ×3 имеют серьезный недостаток, заключающийся в потере связности некоторых фигур. Например, горизонтальный прямоугольник высотой в 2 точки полностью удаляется после такой процедуры. Поэтому на практике приходится разбивать каждую итерацию на несколько подытераций в соответствии с

A	A	A
0	P	0
B	B	B

A	A	A
A	P	0
A	0	2

Рис. 2. Шаблоны конфигурации окрестности удаляемого пиксела p

3	2	1
4	P	0
5	6	7

Рис. 3. Нумерация точек окрестности пиксела p

типом удаляемых контурных точек: для случая 4 подытераций — верхние, левые, нижние и правые точки, для случая 2 подытераций — верхние+левые и нижние+правые точки, или с двойным проходом изображения в шахматном порядке. Согласно классическому алгоритму параллельного утончения Рутовица [4] точка p удаляется, если:

1) B(p)>1;
2) R(p)=2;
3) x(0)×x(2)×x(4)=0 или R(2) ≠ 2;
4) x(6)×x(0)×x(2)=0 или R(0) ≠ 2.

Согласно алгоритму Дойча [4] точка p удаляется, если:

1) R(p)=0,2 или 4;
2) B(p) ≠ 1;
3) x(0)×x(2)×x(4)=0;
4) x(0)×x(2)×x(6)=0;
5) R(p)=4 и ((x(0)×x(6)=1 и x(1)+x(5) ≠ 0 и x(2)+x(3)+x(4)+x(7)=0) или (x(0)×x(2)=1 и x(3)+x(7) ≠ 0 и x(1)+x(4)+x(5)+x(6)=0));

6-8) условия 3-5 после поворота конфигураций на 180 градусов.

Следует отметить, что большинство алгоритмов параллельного утончения имеет те или иные «неудобные» для него фигуры, утончаемые некачественно. Это может быть, например, выражено в сохранении остовов шириной более одной точки, и наоборот, в полном уничтожении некоторых остовов, «разъедании» линий определенной ориентации или квадратов размером 2×2. Такое положение вещей служит побуждающим фактором для создания новых и модификации старых алгоритмов, начиная от добавления специфических условий для оставления точки и кончая вводом конфигураций размером 4×4, 5×5, прямоугольной формы и даже более сложных форм. Сравнительная оценка наиболее известных параллельных алгоритмов утончения приведена в [5], где также предложен новый, по-видимому, один из самых лучших, параллельный алгоритм.

Обособленно от перечисленных алгоритмов стоят неитерационные алгоритмы, которые, как правило, не используют понятия контурных точек и не просматривают их одну за другой, слой за слоем. Например, методы с групповым кодированием, сканируя изображение по линиям в заданном направлении, рассчитывают средние точки для каждого черного сечения произвольной длины, которые и составляют остов. В целях сохранения связности эту процедуру можно повторить в нескольких, обычно перпендикулярных или идущих через 45 градусов направлениях и выбирать в качестве остовных точки, лежащие посередине отрезков наименьшей длины среди отрезков, образующихся в разнонаправленных сечениях изображения. На рисунке 4 представлена фигура, подвергаемая утончению по описываемому методу. Точка B, хотя и лежит в центре горизонтального отрезка линии XX', не является остовной, так как одновременно принадлежит вертикальному отрезку на линии YY', который короче горизонтального и имеет другую центральную точку — точку A, которая и входит в остов.

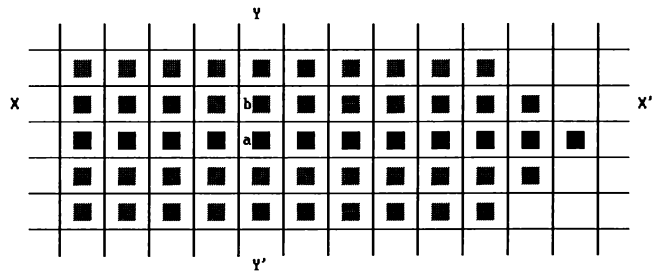


Рис. 4. Нахождение остова с помощью группового кодирования (run-length encoding). Точки фигуры обозначены — ▓, остов — ■.

Также к неитерационным относится эвристический алгоритм. Для его выполнения вводится локальная плотность рисунка

$$d(p) = \sum_{i=0}^{7} x(i)$$

и плотность локальных плотностей

$$D(p) = \sum_{i=0}^{7} d(i)$$

Согласно данному методу точка p остается на изображении, если d(p)<F1 и D(p)<F2 , где F1 и F2 — пороги, значения которых подбираются экспериментально.

Рассмотрение особенностей каждого из приведенных алгоритмов заняло бы слишком много места, поэтому более подробно с ними можно познакомиться, обратившись к первоисточникам. По той же причине за рамками этой работы осталось большое число других рассмотренных в первоисточниках алгоритмов и приемов, таких как детектирование и маркировка значительных выступов в фигурах, перевод окна соседей в одно из значений градаций серого, использование цепных кодов Фримана при трассировке контуров, привлечение различных критериев связности и множественности точек, определение дескрипторов Фурье, введение дополнительной, производной от исходной, сетки изображения и многих, многих других. ■

Литература

1. Ф. Блаум, А. Лейзерсон, Л. Хофстедтер. Мозг, разум и поведение. — М.: Мир, 1988.
2. С. Короткий. Введение в распознавание образов. Часть 1. Преобразование Хафа// Монитор. — 1994. — № 8.
3. Т. Павлидис. Алгоритмы машинной графики и обработки изображений. — М: Радио и связь, 1986.
4. L. Lam, S-W. Lee, C.Y. Suen. Thinning Methodologies — A Comprehensive Survey// IEEE Transactions on Pattern Analysis and Machine Intelligence. — 1992. — Vol. 14. — № 9.
5. B.K. Jang, R.T. Chin. One-Pass Parallel Thinning: Analysis, Properties, and Quantitative Evaluation// IEEE Transactions on Pattern Analysis and Machine Intelligence. 1992. — Vol. 14. — № 11.

Листинг

```
#include <stdio.h>
#include <dos.h>
#include <alloc.h>

typedef unsigned char BYTE;

/* приращение координат X и Y для точек окрестности */
extern int Dirx[8];       /* = {1,1,0,-1,-1,-1,0,1}; */
extern int Diry[8];       /* = {0,-1,-1,-1,0,1,1,1}; */
extern int NG, NV;           /* размеры изображения */
/* массив с ч/б изображением: 1 пиксел - 1 бит      */
extern BYTE far *Image=NULL;
/* массив с пометками на удаление: 1 пиксел - 1 бит,*/
/* признак сохранения - 1, удаления - 0             */
extern BYTE far *Bast=NULL;

/* внешняя функция поворота окрестности точки p на  */
/* 90 градусов, выполняет сдвиг байта окрестности   */
/* на 2 бита                                        */
extern BYTE shiftbyte(BYTE);
/*******************************
_shiftbyte PROC
ARG bb:BYTE
    push bp
    mov bp,sp
    xor ax,ax
    mov al, bb
    rol al,1
    rol al,1
    pop bp
    ret
_shiftbyte ENDP
*******************************/

void ClearBast(void)       /* очистка метки удаления */
{
int i,j;
 for(i=0; i<NG/8; i++)
 for(j=0; j<NV; j++)
 {
  *(Bast+j*(NG/8)+i) = 0xFF;    /* заполнение единицами */
 }
}
/* возвращает значение пиксела (x,y)                */
int image(int x, int y)
{
 BYTE pel;
 pel=*(Image+y*(NG/8)+x/8);
 pel=pel & (1<<(7 - x%8));
 if(pel) return 1;     /* 1(черный) - значимый цвет */
 else    return 0;     /* 0(белый) - пусто          */
}
/* проверка окрестности пиксела (x,y) на соответ-   */
/* ствие конфигурациям; возврат 0 - пометка к       */
/* удалению, 1 - сохранение данной точки            */
int config(int x, int y)
{
 int k,l;
 BYTE byte;

 byte=0;
 for(k=7;k>=0;k--) /* заполнение байта окрестности  */
 {
  byte<<=1;
  byte=byte | image(x+Dirx[k],y+Diry[k]);
 }

 for(k=0;k<4;k++)
 {
  if(k < 2)
  {
   /* две || - конфигурации */
   if((byte & 14) && (byte & 224) && ((byte & 17)==0))
    return 1;
  }
  /* четыре L - конфигурации */
  if((byte & 62) && (byte & 128) && ((byte & 65)==0))
   return 1;
  byte=shiftbyte(byte);       /* поворачиваем на 90 */
 }

 k=0;                 /* проверка на конечную точку */
 for(l=0;l<8;l++) if(byte & (1<<l)) k++;
 if(k<=1) return 1;
 return 0;
}

/* установка метки с (0 - удалить, 1 - сохранить)  */
/* для точки (x,y)                                  */
void SetBast(int x, int y, unsigned c)
{
 BYTE pel;
 pel=*(Bast+y*(NG/8)+x/8);
 pel=((c<<(7 - x%8))) | (pel & ~(1<<(7 - x%8)));
 *(Bast+y*(NG/8)+x/8)=pel;
}

int thinning(void)          /* процедура утончения */
{
int i,j,k;
int active=1;
 ClearBast();
 ShowImage();
 for(;active;)
 {
  active=0;
         /* цикл по правым, верхним, левым и нижним */
                          /* 4-соседям точек (i,j) */
  for(k=0;k<8;k+=2)

  {
   for(i=1;i<NG-1;i++)     /* проход по изображению */
   {
    for(j=1;j<NV-1;j++)
    {
     if(image(i,j)&&(image(i+Dirx[k], j+Diry[k])==0))
     {
      if(config(i,j))
      {
       /* сохранение этого пикселл */
      }
      else
      {
       SetBast(i,j,0);         /* метка на удаление */
       active=1;
      }
     }
    }
   }
          /* удаление помеченных к удалению пикселов */
   for(i=0;i<NG/8;i++)

   {
    for(j=1;j<NV-1;j++)
    {
     *(Image+i+j*(NG/8))=
     *(Image+i+j*(NG/8)) & (*(Bast+i+j*(NG/8)));
     *(Bast+i+j*(NG/8))=0xFF;      /* стирание меток */
    }
   }
  }

  sound(5000);
  delay(100);
  nosound();
 }
 return 0;
}
```

■

Шаблон программ обработки файлов

С. Шмаков

Нахожу на винчестере незнакомую (или забытую?) программу. Что она делает? Друг, сидящий рядом, тоже не знает. «А ты запусти ее», — советует он. Так и поступаю, предварительно приняв меры антивирусной предосторожности. Смотрим на экран.

Бывает, весь он окрашивается приятным для глаз цветом, появляется меню или указание, как его вызвать, в окне красуется заголовок программы и сведения о ее авторе, звучит музыкальная заставка. Так, значит, это программа диалоговая. Если из заголовка не вполне ясно ее назначение, можно вызвать HELP (обычно клавишей F1), пройтись по меню. Выбрав пункт Exit (Выход), видим, что первоначальное состояние экрана восстановилось. Это характерная черта диалоговых программ. Работать с ними, особенно с русифицированными, приятно, но объем на диске занимается приличный.

В других случаях на экране высвечивается заставка и инструкция по формату вызова программы, длинная или короткая. Программа после этого завершается сама. Она, стало быть, относится к разряду командностроч-ных, то есть таких, вся информация для работы которых (имена файлов, команды, ключи) указывается в командной строке, а в дальнейшем при необходимости происходит лишь лаконичный диалог в стиле: «Сделать ли так-то?» — «Да», «Нет». Так работают не только мелкие утилиты, но и весьма уважаемые программы с широким набором функций (компилятор командной строки TCC.EXE, архиваторы, ассемблеры, редакторы связей, компиляторы Fortran-77). Развитый диалоговый режим утяжелил бы их до невозможности. Но, конечно, командно-строчный стиль более характерен именно для небольших утилит.

Такие модули обработки файлов частенько пишут программисты для своих собственных нужд. Вот, например, выдержка из описи моего арсенала.

1. Адаптация текстового файла к требованиям редакции журнала «Монитор» (удвоение некоторых символов, исключение лишних пробелов и знаков табуляции и переносов, ограничение длины строки).
2. Замена цепочек пробелов на знаки табуляции и наоборот.
3. Подсчет числа печатных знаков и авторских листов в текстовом файле.
4. Перекодировка текстов из альтернативной системы в базовую и наоборот.
5. Обработка листингов SOURCER (исключение лишнего).
6. Очистка текстовых файлов от управляющих кодов принтера [1].
7. Замена символьной комбинации «((0))» на двоичный нуль [1].

И каждый раз приходится наряду со специфическими вопросами обработки файлов решать вопросы повторяющиеся, как-то:

а) что нужно сделать с файлом, если утилита полифункциональная, то есть, какие ключи предшествуют его имени;
б) есть ли файл с таким именем на диске;
в) будет ли выходной файл, как он должен называться, не перекрывает ли он уже существующий файл и удалось ли ему уместиться на диске;
г) успешно ли закончилась обработка исходного файла;
д) следует ли выдать подсказку для пользователя.

Все это наводит на мысль о написании некоего общего шаблона, который занимался бы всем этим и подносил бы программисту «на блюдечке» указатель входного (и, возможно, выходного) потока, давая возможность сосредоточиться на конкретных специфических вопросах обработки данных. Выберем для этого наиболее знакомый программисту язык, на котором будут в дальнейшем написаны индивидуальные части программ. В данном случае это язык С. Приверженцы других языков имеют возможность перевести этот шаблон на них и считаться авторами.

Как должен происходить обмен данными между индивидуальной частью программы и шаблоном?

Условимся сначала, что тело шаблона не будет модифицироваться напрямую, а будет подключаться директивой #include. Обмен данными может происходить двумя путями: а) определение переменных директивами #define, что повлечет за собой препроцессорное модифицирование тела шаблона; б) вызов функции с соответствующими аргументами. Первый путь предпочтительнее: здесь еще на этапе компиляции отсекаются части шаблона, не нужные в данном случае, выявляются несоответствия между передаваемыми данными. Сокращается размер объектного кода, программы реже выходят за пределы 64 килобайтов, допустимых для COM-формата. Когда для передачи данных используется лишь этот путь, программа состоит из двух функций: main (шаблон, модифицированный препроцессором) и file_process (индивидуальная функция обработки данных, на вход поступают указатели на входной-выходной потоки, возвращаемое значение — признак успешного или неуспешного завершения обработки данных).

Второй путь единственно возможен для передачи массивных данных. В нашем случае — это набор ключей (символьных строк), каждый из которых может предшествовать имени обрабатываемого файла в командной строке, если утилита полифункциональная. Предполагается, что ключей может быть не более 16. План такой утилиты следующий: main (определяется массив ключей и передается в вызываемую функцию file_manager), file_manager (шаблон, модифицированный препроцессором или преобразует ключи, встречающиеся, отсутствующие в командной строке, в двухбайтовое слово, каждый бит которого, начиная с младшего, отвечает за наличие своего ключа: 1 — есть ключ, 0 — нет), file_process (на входе принимает указатели на входной-выходной потоки и упомянутое слово состояния ключей).

Текст описываемого шаблона вы найдете в листинге 1, а листинги 2 и 3 иллюстрируют его применение (2 — перевод текстовых файлов из альтернативной кодировки в базовую, 3 — полифункциональная утилита сжатия/распаковки текстовых файлов путем замены пробелов на знаки табуляции и наоборот, ключ '-' перед именем обрабатываемого файла означает его распаковку). ■

Литература

1. С. Шмаков. «Дирижерская палочка» для принтера// Монитор. — 1994. — № 7.

Листинг 1

```
/* Шаблон программ обработки файлов */
#if !defined(IN_TYPE)
  #error Не определена константа IN_TYPE
#endif
#if defined(OUT_TYPE)
  #include <dir.h>
#elif defined(NEW_EXT)
  #error Константа NEW_EXT определена, а OUT_TYPE нет
#endif
#if defined(N_keys)
  #if N_keys<1 :: N_keys>16
    #error Недопустимое значение константы N_keys
  #else
    #include <string.h>
  #endif
#else
  #if !defined(HEADER)
    #error Не определены константы N_keys и HEADER
  #endif
#endif
#include <stdio.h>
void _setenvp(){}
void
#if defined(N_keys)
    file_manager(int argc,char *argv[],char *keys[])
#else
    main(int argc,char *argv[])
#endif
{
  int n; FILE *in;
#if defined(N_keys)
  unsigned int bit_key=0,mask; int i;
#endif
#if defined(OUT_TYPE)
  #if !defined(NEW_EXT)
  char fn_bak[MAXPATH];
  #endif
  char drive[MAXDRIVE],dir[MAXDIR],name[MAXFILE],
       ext[MAXEXT],fn_out[MAXPATH];
  int out_flag; FILE *out;
  int
#else
  void
#endif
      file_process(FILE *
#if defined(OUT_TYPE)
                          ,FILE *
#endif
#if defined(N_keys)
                                 ,unsigned int
#endif
                 );
#if !defined(N_keys)
  printf(HEADER);
#endif
  for (n=1; n<argc; ++n)
  {
#if defined(N_keys)
  for (mask=1,i=0;
    i<N_keys && strcmp(argv[n],keys[i]);mask<<=1,++i);
  if (i<N_keys) bit_key|=mask;
  else
#endif
  if ((in=fopen(argv[n],IN_TYPE))!=NULL)
  {
#if defined(OUT_TYPE)
    fnsplit(argv[n],drive,dir,name,ext),
  #if defined(NEW_EXT)
```

```
      fnmerge(fn_out,drive,dir,name,NEW_EXT);
  #else
      fnmerge(fn_out,drive,dir,name,".$$$"),
      fnmerge(fn_bak,drive,dir,name,".bak");
  #endif
      if ((out=fopen(fn_out,OUT_TYPE))!=NULL)
#endif
      {
#if defined(OUT_TYPE)
        out_flag=
#endif
              file_process(in
#if defined(OUT_TYPE)
                           ,out
#endif
#if defined(N_keys)
                               ,bit_key
#endif
                          ),
        fcloseall();
#if defined(N_keys)
        bit_key=0;
#endif
#if defined(OUT_TYPE)
        if (out_flag==EOF)
          printf("\nОшибка при записи файла %s",
          fn_out), remove(fn_out);
  #if !defined(NEW_EXT)
        else if (rename(argv[n],fn_bak))
          printf("\nФайл %s уже существует!",fn_bak);
        else if (rename(fn_out,argv[n]))
          printf("\nОстался файл %s",fn_out);
  #endif
#endif
        printf("\nЗакончена обработка файла %s",
        argv[n]);
      }
#if defined(OUT_TYPE)
      else printf("\nФайл %s не удалось создать!",
        fn_out), fcloseall();
#endif
    }
    else
      printf("\nФайл %s не удалось открыть!",
      argv[n]);
  }
#if defined(HELP)
  if (argc==1) printf(HELP);
#endif
}
```

■

Листинг 2

```
#define HEADER "\nПерекодировка из альтернативной \
таблицы в базовую."
#define IN_TYPE "rt"
#define OUT_TYPE "wt"
#define HELP "\nФормат вызова: alt2bas <входной файл>\
 [...]\n"
#include "filman.c"
int file_process(FILE *in,FILE *out)
{   static unsigned char newsym[48]={
    155,156,157,165,167,131,132,133,
    134,151,149,145,146,139,140,161,
    163,168,166,169,164,170,141,142,
    147,144,152,150,153,148,154,128,
    129,130,135,136,137,138,143,158,
    159,162,160,171,172,173,174,175};
  unsigned char a[250]; int i,out_flag;

  while (fgets(a,250,in)!=NULL)  /* Цикл по строкам */
  {   for (i=0; a[i]; ++i)
      if (128<=a[i] && a[i]<=175) a[i]+=48;
      else if (176<=a[i] && a[i]<=223)
        a[i]=newsym[a[i]-176];
    if ((out_flag=fputs(a,out))==EOF) break;
  }
  return out_flag;
}
```

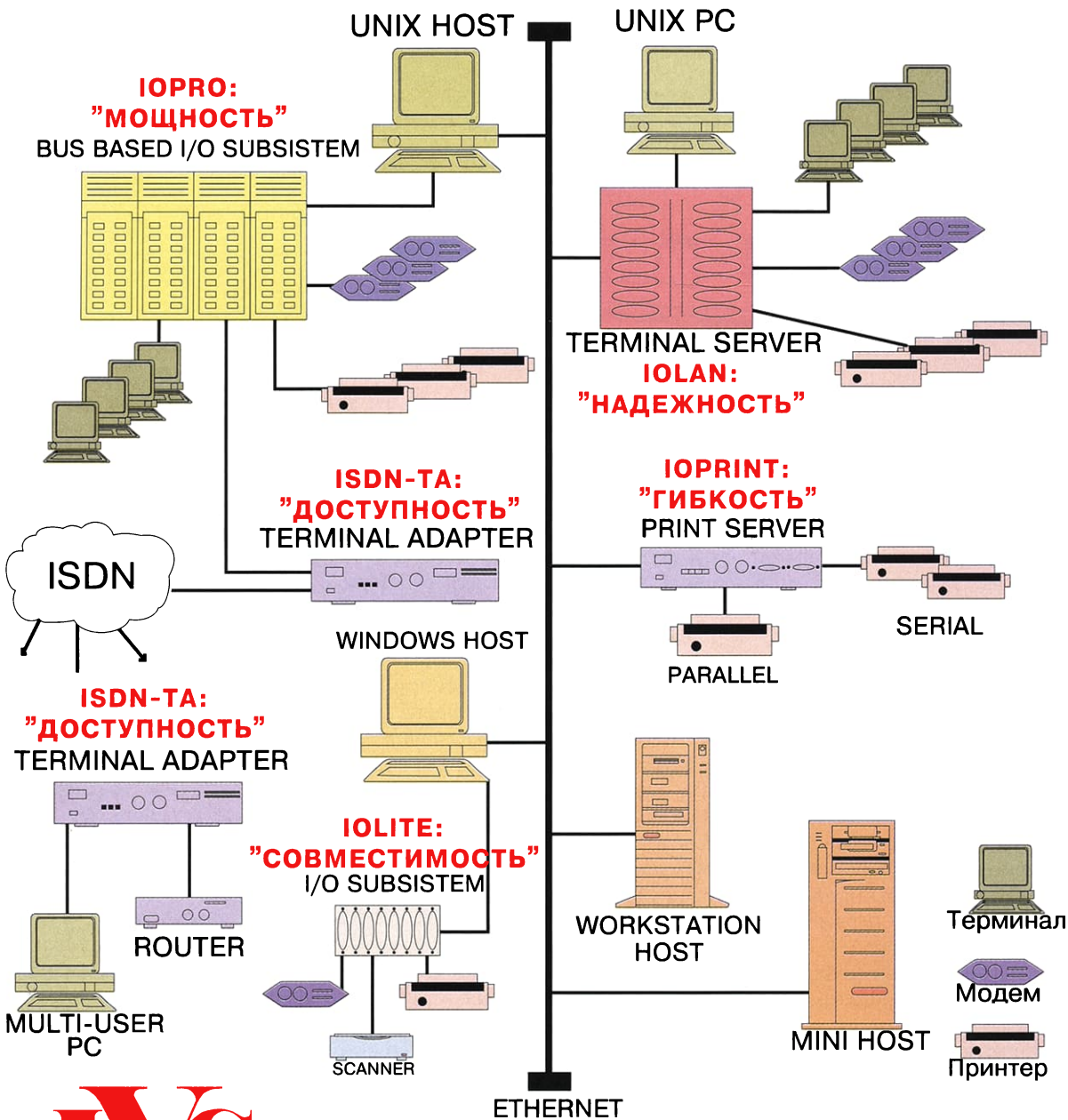
■

Листинг 3

```
#define IN_TYPE "rt"
#define OUT_TYPE "wt"
#define N_keys 1
#define HELP "\nФормат вызова: tabs [-] <входной файл>\
  [...]\n\                  Минус означает "наоборот"."
#include "filman.c"
void main(int argc,char *argv[])
{ static char *keys[N_keys]={"-"};
  printf("\nЗамена пробелов на знак табуляции и наоборот"),
  file_manager(argc,argv,keys);
}
int file_process(FILE *in,FILE *out,unsigned bit_key)
{ char a[250],b[250],apo;
  int i,pos,j,N_blanks,N_crlf,out_flag;

  N_crlf=apo=0;
  while (fgets(a,250,in)!=NULL) /* Цикл по строкам */
  {
    for (i=pos=j=N_blanks=0; a[i]!='\n'; ++i)
    {
      if (!bit_key && !(pos%8))
      { if (N_blanks>1 && !apo) b[j++]='\t';
        else while (N_blanks--) b[j++]=' ';
        N_blanks=0;
      }
      else if (N_blanks && a[i]!=' ' && a[i]!='\t')
        do b[j++]=' '; while (--N_blanks);
      if (a[i]==' ') ++N_blanks, ++pos;
      else if (a[i]=='\t') N_blanks+=(8-(pos%8)),
        pos=(pos/8)*8+8;
      else
      { ++pos, b[j++]=a[i];
        if (a[i]=='\'' :: a[i]=='\"')
        { if (!apo) apo=a[i];
          else if (apo==a[i]) apo=0;
        }
      }
    }
    while (j && (b[j-1]==' ' :: b[j-1]=='\t')) --j;
    if (j)
    { b[j]='\n', b[j+1]=out_flag=0;
      while (N_crlf && (out_flag=fputs(13,out))!=EOF)
        --N_crlf;
      if  (out_flag==EOF ::
          (out_flag=fputs(b,out))==EOF)
        break;
    }
    else ++N_crlf;
  }
  return out_flag;
}
```

■

Корпорация LVS предлагает экономичные сетевые решения фирмы CHASE RESEARCH



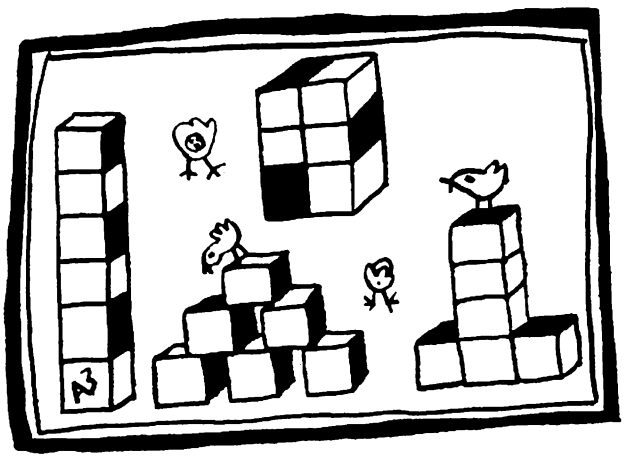
LVS CORPORATION

SCO OK

117342, Российская Федерация,
Москва, Профсоюзная ул., 65
Тел.: (095) 330-42-66, (095) 330-16-06
Факс: (095) 330-36-93
E-mail: sco@lvs.msk.su

Авторизованный системный интегратор

Гибкая статическая компоновка



С. Короткий

С первых дней развития компьютерной техники перед программистами стоит проблема минимизации объема исполняемых кодов. Новейшие достижения в технологии производства аппаратных средств снизили ее остроту, однако, каким бы мощным ни был компьютер, программа должна использовать его ресурсы эффективно, не допуская неоправданных затрат оперативной памяти и не уповая на быстроту современных процессоров. Это становится особенно очевидно в связи с появлением новых классов задач — «монстров», препоручаемых компьютеру, таких, например, как синтез объемных динамических изображений в реальном времени.

Для уменьшения размеров загрузочных модулей разработано несколько приемов. Во-первых, это использование оптимизирующих компиляторов, генерирующих компактный код. Во-вторых, построение программ с оверлейной структурой. В-третьих, с появлением Windows возникла возможность создавать для приложений Windows библиотеки с динамической компоновкой. Второй и третий способы имеют ограниченную область применения, так как первый из них неприемлем для среды Windows, а второй — для среды DOS. Первый способ ненадежен с той точки зрения, что далеко не все оптимизированные программы работают так, как это подразумевал программист, а степень сжатия грамотно написанного кода невелика. Существует, однако, еще один, значительно менее известный прием, позволяющий при статической компоновке уменьшить объем программы. Он с разной эффективностью применим к различным программам, но в некоторых случаях способен дать большой выигрыш.

Предположим, имеется некая библиотека, входящая в состав программы. Если ее нет, ее всегда можно создать из объектных файлов, участвующих в проекте. Обычно в целях достижения оптимальной компоновки EXE-файла исходные модули составляют с таким расчетом, чтобы при включении модуля в файл как можно больший процент входящих в модуль процедур имел ссылки на себя из других мест программы. Очевидно, что самым простым и безотказным способом достичь этого является помещение каждой процедуры в отдельный исходный модуль, что часто применяется в коммерческих библиотеках. При этом компоновщик помещает в исполняемый модуль исключительно функции, на которые есть ссылки. Однако возникает вопрос: все ли ссылки работают в процессе выполнения программы? Для довольно простых программ ответ, как правило, однозначен: да. Среди крупных же проектов очень часто встречается обратная ситуация. Например, большинство библиотек интерфейсных функций имеет в своем арсенале процедуры для поддержки различных экранных управляющих элементов. Ядро библиотеки, остающееся всегда без изменений, выполняет все операции по настройке и обслуживанию этих элементов. Это означает, что вне зависимости от того, использует ли данная программа тот или иной тип управляющего элемента, некоторые или даже большинство подпрограмм его поддержки будут включены в EXE-файл. В связи с этим в некоторых библиотеках, поставляемых в исходных кодах, вводятся специальные определения для препроцессора, чтобы обеспечить условную компиляцию. Однако далеко не все авторы программ охотно раздают, пусть и за деньги, свои исходные коды, так как они несут в себе не только алгоритмические реализации, но и «know how» — идеологию программирования. Кроме того, перекомпилировать библиотеку каждый раз очень накладно. Для преодоления этих проблем предлагается следующий подход, который будет проиллюстрирован примерами на языке C.

Каждая из подпрограмм, подключаемых в EXE-файл по требованию, описывается как указатель на функцию. Пусть, например, в программе необходима функция построения на экране прямоугольника. Ее обычный прототип

```
void rectangle(int x, int y, int dx, int dy, int colour);
```

заменяется на

```
void (*rectangle)(int x, int y, int dx, int dy, int colour);
```

Прежняя функция рисования прямоугольника заново обозначается, например, так

```
void rect_1(int x, int y, int dx, int dy, int colour);
```

Теперь внутри программы вместо вызовов функции по ее адресу стоят косвенные вызовы через указатель. Прямых ссылок на функцию нет, и она будет отсутствовать в EXE-файле до тех пор, пока программист не сделает присваивание

```
rectangle=rect_1;
```

В этом случае соответствующая функция подключится компоновщиком. Кроме того, в ядро библиотеки вводится функция-заглушка

```
int dummy(void){return 0;}
```

Она дает возможность присвоить указателю rectangle осмысленный адрес даже в случае, когда функция рисования не нужна.

```
rectangle=dummy;
```

Это позволяет избежать неприятностей при вызове функции по неосторожности. Как известно, в C за очисткой стека после вызова подпрограммы следит вызывающая функция, поэтому наличие прототипа указателя делает возможным несоответствие числа аргументов у функции-заглушки и заменяемой ею функции. Функция-заглушка возвращает ноль — признак успешного завершения — для тех случаев, когда заменяемая ею функция должна возвращать код ошибки.

Рассмотрим некую библиотеку оконных функций. Вот примерный фрагмент кода из нее, иллюстрирующий постоянную привязку функции rectangle в ядре библиотеки до ее модификации.

```
for(i=0;i<NumOfWindows;i++) /* перебираем все открытые окна   */
{
   if(IsActive(i))        /* ищем активное в настоящий момент */
  {
   rectangle(WorkZone[i].x,WorkZone[i].y,  /* обновляем его  */
   WorkZone[i].dx,WorkZone[i].dy, colour); /* рабочую зону   */
  }
 }
```

После замены функции на указатель, и может быть — не одной, в библиотеку вводятся или дополняются уже существующие функции инициализации работы с управляющими элементами, например,

```
int InitWindows(void)
{
/*определение других указателей и прочая настройка */
 ........
 rectangle=rect_1;
}
```

По умолчанию указатели на все функции инициализируются адресом функции-заглушки.

```
void (*rectangle)(int,int,int,int,int)=dummy;
```

В заголовочный файл библиотеки lib.h вводятся макроопределения, облегчающие инициализацию системы. Вот один из возможных вариантов.

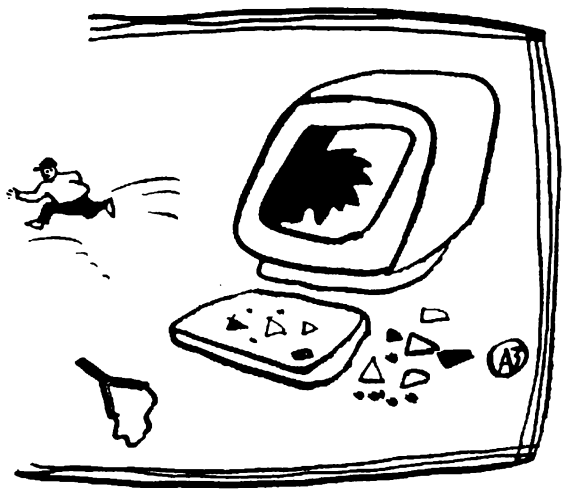
```
#ifndef UseWindows
#define InitWindows();
#endif
#ifndef UseButtons
#define InitButtons();
#endif
#ifndef UseScrolls
#define InitScrolls();
#endif
#ifndef UseSwitches
#define InitSwitches();
#endif


#define InitAllControls();InitWindows();InitButtons();\
InitScrolls(); InitSwitches(); ..........
```

Тогда в исходном модуле, производящем настройку библиотеки, достаточно написать:

```
#include <stdio.h>                /*подключение стандартных */
                                  /* заголовочных фалов     */
...
#define UseWindows /*определяем, что собираемся использовать */
#define UseButtons
...
#include «lib.h»   /* подключаем только используемые функции */
main()
{  ...
InitAllControls();


}
```

Таким образом, предложенный метод дает возможность избавиться от косвенных расходов памяти на «слепой» программный код, который присутствует во многих программах со сложной управляющей структурой. Данный подход позволяет также эмулировать подгрузку драйверов различных устройств и режимов работы. Например, кроме функции rect_1, работающей в графическом режиме А, может быть rect_2 для графического режима Б. При этом варианты функции для режимов, к которым программист не собирается адаптировать данную программу, не вводятся в загрузочный файл. ■

Событийно-управляемый интерфейс в СУБД FoxPro



А.Дубина, С.Сынгур

FoxPro поставляется с двумя примерами построения приложений, работающих с несколькими файлами. Примеры расположены в каталоге GOODIES/FNDATION и называются EX1 и EX2. Обе программы позволяют работать с несколькими активными окнами, переключаясь между ними с помощью «мыши» или из меню, которое также доступно. Организация такого интерфейса с пользователем стала возможной с появлением в FoxPro Foundation READ — нового средства, позволяющего вызывать любой режим программы из любого места, пока READ ожидает действий пользователя. В своей книге «FoxPro 2.0 Applications Programming» Лес Пинтер приводит интересный и более доступный для понимания простых пользователей пример написания подобных программ. Однако, кроме всего перечисленного, программы EX1 и EX2 обладают еще одним неоценимым свойством. Сколько бы мы ни вызывали на выполнение приложений, управляющая программа организована таким образом, что глубина вложенности DO для вызова исполняемой программы не превышает 3-4 в состоянии ожидания события. В этом можно убедиться, исследовав работу программы функцией SYS(16). Такой метод управления подпрограммами позволяет иметь на экране сколько угодно активных приложений, работать с ними, передавая управление с помощью «мыши». Если в программе Леса Пинтера ABSTRACT реализована возможность вызова любого режима из любого места программы, а программа EX1(EX2) работает с несколькими активными базами, то предлагаемая программа EX1M позволяет одновременно работать с несколькими независимыми программами-приложениями. Все программы-приложения являются самостоятельными объектами, состоящими из среды данных и интерфейсной части, обеспечивающей связь пользователя с программой. Главная программа — объект, управляющий работой программ-приложений. Практически речь идет о реализации интерфейса типа «рабочий стол» стандартными средствами FoxPro.

Структура программы приведена на рис. 1.

В качестве примера пиведем программу EX1M, которая является модернизацией программы EX1, поставляемой с FoxPro.

Листинги 1 и 2 могут быть использованы для написания рабочих приложений. При данном методе управления каждое приложение является самостоятельным объектом, состоящим из среды данных и интерфейсной части, обеспечивающей связь пользователя с программой. Для регистрации объекта в системе достаточно в программе «myhandler» в цикле DO CASE добавить две строки:

```
CASE WONTOP(«P1»)
     DO p1.spr
```

Здесь P1 — имя окна объекта, а p1.spr — имя файла, содержащего код программы, реализующей объект. Необходимо также в программе «efface» в строке

```
IF INLIST(m.x,«P1»,«P2»)
```

в список имен окон дописать имя окна нового объекта. Такие несложные действия позволяют легко менять конфигурацию среды. Можно добавлять или удалять приложения по усмотрению разработчика. Становится возможной сборка процессов из объектов. И это делается так же легко, как маленькие дети из кубиков строят сказочные крепости. ■

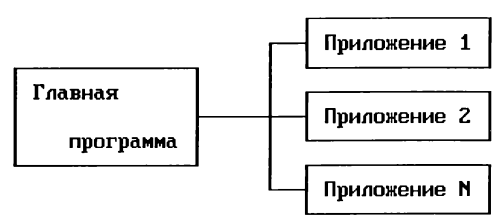


Рис. 1

Листинг 1

```
*                  Главная программа                  *
PROC ex1m
PUBLIC dropdead,tobedone
SET TALK OFF
dropdead = .F.
tobedone=""
PUSH MENU _MSYSMENU
SET SYSMENU TO
SET SYSMENU AUTOMATIC
DEFINE PAD _qnx0tyxd5 OF _MSYSMENU PROMPT "Раздел 1" ;
COLOR SCHEME 3
DEFINE PAD _qnx0tyxfz OF _MSYSMENU PROMPT "Раздел 2" ;
COLOR SCHEME 3
ON PAD _qnx0tyxd5 OF _MSYSMENU ACTIVATE POPUP _qnx0t3450
ON PAD _qnx0tyxfz OF _MSYSMENU ACTIVATE POPUP _qnx0tee4z
DEFINE POPUP _qnx0t3450 MARGIN RELATIVE SHADOW ;
COLOR SCHEME 4
DEFINE BAR 1 OF _qnx0t3450 PROMPT "Пункт 1"
DEFINE BAR 2 OF _qnx0t3450 PROMPT "Пункт 2"
DEFINE BAR 3 OF _qnx0t3450 PROMPT "Выход  "
ON SELECTION BAR 1 OF _qnx0t3450 DO mhit WITH "P1.spr"
ON SELECTION BAR 2 OF _qnx0t3450 DO mhit WITH "P2.spr"
ON SELECTION BAR 3 OF _qnx0t3450 DO QT
READ VALID myhandler()
CLEAR WINDOW ALL
CLOSE DATABASES
SET TALK ON
POP MENU _MSYSMENU
PROCEDURE mhit
PARAMETER prog
IF RDLEVEL() > 1
   tobedone = prog
   CLEAR READ
ELSE
   DO (prog)
ENDIF
PROCEDURE QT
dropdead = .T.
CLEAR READ ALL
FUNCTION myhandler
PRIVATE m.temp, m.x
IF dropdead
   RETURN .T.
ENDIF

IF LEN(tobedone) > 0
   m.temp = tobedone
   tobedone = ""
   DO (m.temp)
   RETURN .F.
ENDIF

DO CASE
CASE WONTOP("P1")
   DO p1.spr
CASE WONTOP("P2")
   DO p2.spr
ENDCASE
RETURN .F.
PROCEDURE efface
PRIVATE m.x
m.x = WCHILD("",0)
DO WHILE LEN(m.x) > 0
   IF INLIST(m.x,"P1","P2")
      RETURN
   ENDIF
   m.x = WCHILD("",1)
ENDDO
FUNCTION stopread
PARAMETER m.window
IF NOT WVISIBLE(m.window)
   quitting = .T.
ENDIF
RETURN quitting OR NOT WREAD()
```

Листинг 2

```
* 06/30/94 P1.SPR 09:37:59 Программа - приложение   *
#REGION 0
REGIONAL m.currarea, m.talkstat, m.compstat
IF SET("TALK") = "ON"
        SET TALK OFF
        m.talkstat = "ON"
ELSE
        m.talkstat = "OFF"
ENDIF

m.compstat = SET("COMPATIBLE")
SET COMPATIBLE FOXPLUS

IF NOT WEXIST("P1")
DEFINE WINDOW P1 ;
FROM INT((SROW()-11)/2),INT((SCOL()-46)/2) ;
TO INT((SROW()-11)/2)+10,INT((SCOL()-46)/2)+45 ;
TITLE "P1" FLOAT CLOSE SHADOW DOUBLE COLOR SCHEME 1
ENDIF
#REGION 1
REGIONAL p_P1
PRIVATE quitting
quitting = .F.
IF USED("gr")
        SELECT gr
        SET ORDER TO npp
ELSE
        SELECT 0
        USE (LOCFILE("gr.dbf","DBF","Where is gr ?"));
                AGAIN ALIAS gr ORDER npp
ENDIF
#REGION 1
IF WVISIBLE("P1")
        ACTIVATE WINDOW P1 SAME
ELSE
        ACTIVATE WINDOW P1 NOSHOW
ENDIF

@ 8,0 GET p_P1 ;
  PICTURE "@*HN Начало;Конец;Вперед;Назад;Выход" ;
  SIZE 1,8,1 DEFAULT 1 VALID _qrb0knbn9()
@ 0,1 GET gr.npp SIZE 1,4 DEFAULT " " DISABLE
@ 0,6 GET gr.name SIZE 5,37 DEFAULT " "
@ 5,0 TO 5,43
@ 7,0 TO 7,43
@ 6,3 SAY "Отметка на удаление :"
@ 6,28 SAY IIF(deleted(),'Удалить','       ') SIZE 1,7
```

```
  IF NOT WVISIBLE("P1")
          ACTIVATE WINDOW P1
  ENDIF
  READ CYCLE ;
        ACTIVATE _qrbOknch7() DEACTIVATE stopread('P1')
  #REGION 0
  IF m.talkstat = "ON"
          SET TALK ON
  ENDIF
  IF m.compstat = "ON"
          SET COMPATIBLE ON
  ENDIF
  #REGION 1
  IF quitting
     RELEASE WINDOW P1
     pack
     DO efface
  ENDIF
  FUNCTION _qrbOknbn9     &&  p_P1 VALID
  #REGION 1
  DO CASE
  CASE p_P1 = 1
     GO TOP
  CASE p_P1 = 2
     GO BOTTOM
  CASE p_P1 = 3
     IF EOF()
        GO BOTTOM
     ELSE
        SKIP
        IF EOF()
           GO BOTTOM
        ENDIF
     ENDIF
  CASE p_P1 = 4
        IF BOF()
        GO TOP
     ELSE
        SKIP -1
        IF BOF()
           GO TOP
        ENDIF
     ENDIF
  CASE p_P1 = 5
     quitting = .T.
     CLEAR READ
  ENDCASE
  @ 6,28 say IIF(deleted(),'Удалить','       ') SIZE 1,7
  SHOW GETS
  RETURN 0
  FUNCTION _qrbOknch7     && Read Level Activate
  #REGION 1
  sele gr
  show gets                                              ■
```

АНОНС

«Монитор» — 2.95

А. Козленков.
IBM OS/2 Warp — музыка атаки

С. Короткий.
Введение в распознавание образов. Часть 3. Выделение углов

А. Тихонов.
Система программирования Stony Brook Pascal+

В. Плотников.
Использование возможностей защищенного режима работы процессора в Windows-приложениях

С. Ерёмин.
Еще раз о TSR...

А. Дубина, С. Сынгур.
Процедурные файлы — платформа для реализации полиморфизма в среде xBase

С. Миллер.
Организация псевдомногозадачности в Turbo Vision

А. Гончаров.
Звездные войны от «Silmarils»

М. Трофимов.
Жизнь на Макинтоше: по страницам новой книги. Часть 1

А. Горчаков.
Компьютерные решения в гуманитарном образовании

В. Дьяконов.
Базовые средства мультимедиа в Microsoft Windows

А. Боресков, Е. Шикин.
3D Studio: работа с текстурой

В. Дьяконов.
MathCAD 4.0 — что там новенького?

А. Маркин.
Утечка информации за счет побочных излучений компьютера

Журнал «Монитор» можно приобрести в городах России, Украины, Белоруссии, Узбекистана, обратившись к нашим официальным дилерам.

1. Баку
Фирма «Марко», ул.Хагани, 33, тел. (8922) 98-90-82.

2. Владивосток
Фирма «Компьютерные технологии», тел. (4232) 221-291.

3. Волгоград
Арендное предприятие «Техническая книга», тел. (8442) 363-597.

4. Днепропетровск
Попов А.Е., тел. (0562) 428-593.

5. Железноводск
Фирма «Кросс, Лтд.», тел. (6533) 99-563, 94-468.

6. Иркутск
Мадьяров Игорь Иосифович, тел. (3952) 46-19-68

7. Киев
Фирма «Диалектика», тел. (044) 266-40-74; Фирма «ОКС», тел. (044) 446-95-05.

8. Минск
НПП «Триумф», тел. (0172) 661-609, 661-680, факс 666-606.

9. Пермь
Попов Андрей Леонидович, тел. (3422) 33-30-91

10. Ростов-на-Дону
Дмитрий Лагунов, тел. (8632) 24-13-10.

11. Санкт-Петербург
Дом книги (812) 219-94-57; Магазин «Техническая книга» (812) 164-65-65.

12. Тамбов
АО «От Елены» ЛПС «Shift», тел. (0752) 22-44-63.

13. Ташкент
Центр молодежных исследований, тел. (3712) 688-847.

14. Уфа
ПКФ «Шамиль», тел. (3472) 23-79-78.

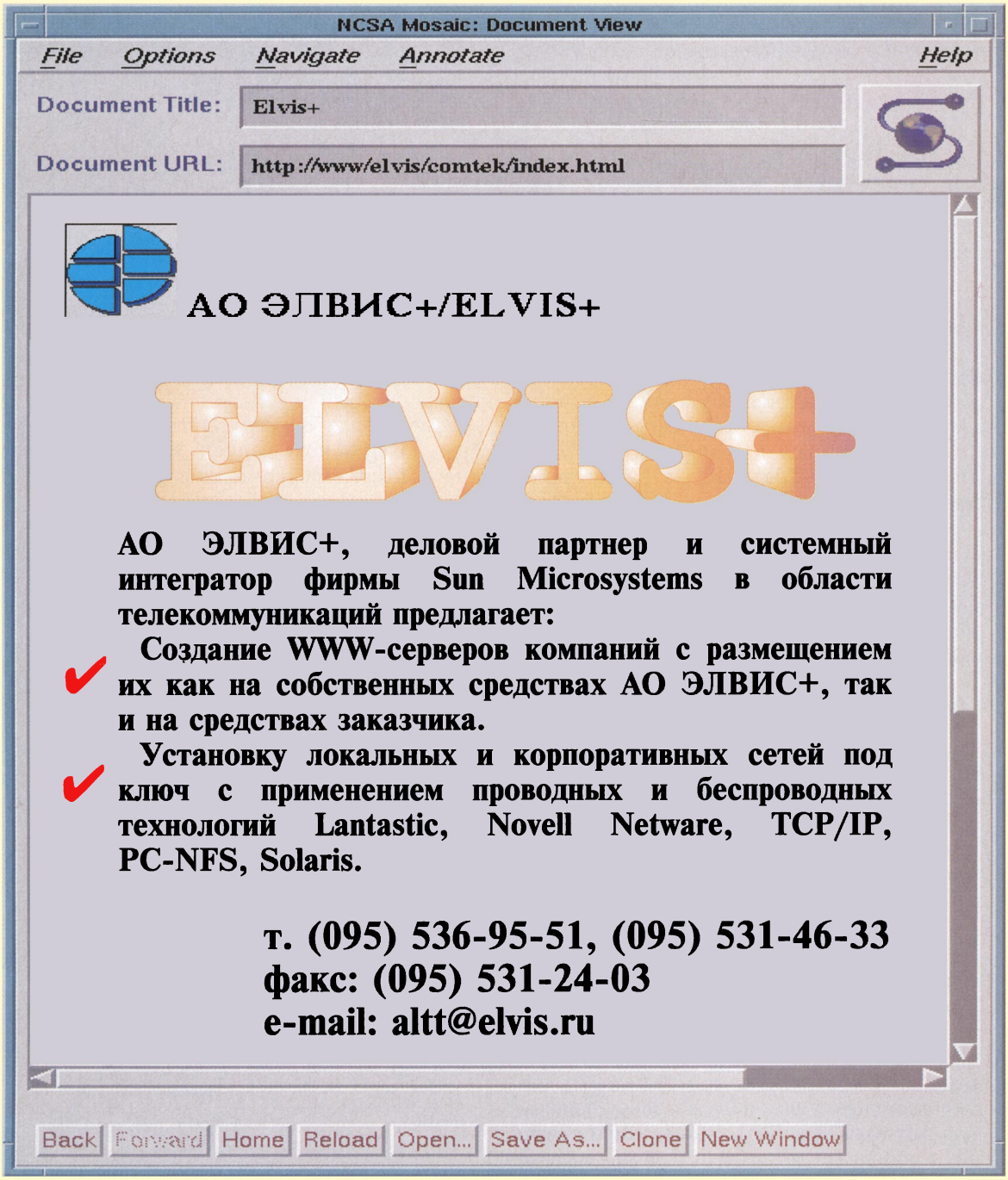
15. Харьков
ЧНПФ «Харьков», тел. (0572) 23-37-22, факс 44-10-00.

16. Чебоксары
Фирма «Демос», тел. (8352) 208-395, 426-492, факс 209-437.

Приглашаем дилеров, тел. (095) 238-99-11.

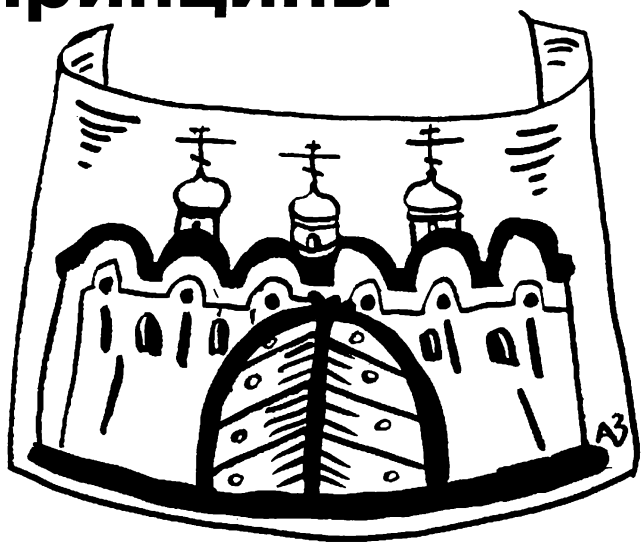
Что такое WWW ?

Это окно в мир, через которое Ваша реклама может быть доступна 30 миллионам пользователей сети Internet. Вид рекламы самый разнообразный: графика, аудио, видео). Несколько тысяч ведущих фирм мира уже открыли свои WWW-серверы.



СУБД HyTech 2.0: архитектура и принципы работы

С. Ковалевский,
А. Малярский,
К. Зайцев, С. Ушенин,
В. Нечаев, В. Хруст



1. Что такое СУБД HyTech?

СУБД HyTech — это совокупность программных средств, предназначенных для создания, ведения и использования реляционных БД. Она позволяет пользователям обращаться к данным с помощью специальных языков SQL и QBE, а также через Call-интерфейс из языков программирования C и Pascal. В состав СУБД HyTech входят средства разработки пользовательских приложений HyTech_Vision и HyTech_Designer, средства просмотра и редактирования баз данных HyTech_Browse, интегрированная среда с языками SQL и QBE, библиотеки функций для языков программирования C и Pascal, интерпретатор языка HL_SQL, SQL-сервер баз данных HyTech.

2. Объекты баз данных, поддерживаемых HyTech

Поля, входящие в запись, могут быть как обычными, так и ключевыми (для ускорения операций поиска). Для каждого ключевого поля строится индекс. Ключ может быть первичным (все значения которого в пределах таблицы уникальны) и вторичным (допускается повтор значений).

Несколько символьных полей, расположенных смежно, могут рассматриваться как одно ключевое поле — *группа.*

Некоторая часть любого символьного поля может рассматриваться как ключевое поле — *подстрока.* Подстрока обязательно должна быть *ключом*, для которого строится *индекс.* По его значениям производится поиск.

Атомарная составляющая записи может рассматриваться как столбец таблицы. Содержит определенную характеристику записи. Имеет имя, длину и тип.

Число байтов, отводимых для хранения данных, содержащихся в поле записи: длина поля зависит от типа и может варьироваться в пределах от 1 до 65517 байтов.

Алфавитно-цифровой идентификатор поля записи имеет длину до 10 символов (русского или латинского алфавита) и должен быть уникальным для таблицы.

3. Типы данных и структура файлов HyTech

Поддерживаются следующие типы полей (табл. 1).

Таблица 1

Тип	Длина	Возможные значения	Запрещенные значения
ARRA	1-65517	0 : 254	255(*)
CHAR	1-65517	1 : 254	255(*), 0(**)
CURR	8	1.7E-308 : 1.7E+308	+NAN/-NAN(***)
BYTE	1	0 : 254	255
DATE	2	1 : 65379	65535
DFLT	8	1.7E-308 : 1.7E+308	+NAN/-NAN(***)
DWRD	4	0 : 4294967294	4294967295
FLOA	4	3.4E-38 : 3.4E+38	+NAN/-NAN(***)
INTR	2	-32767 : +32767	-32768
LONG	4	-2147483647 : +2147483647	-2147483648
NMBR	3	0 : 16777214	16777215
WORD	2	0 : 65534	65535

СУБД HyTech обеспечивает следующие *характеристики*:
- количество записей в одном файле — до 16 777 215,
- длина записи — до 65 517 байтов,
- количество полей в записи — до 8192,
- длина ключевого поля — до 255 байтов,
- длина простого поля — до 65 517 байтов,
- количество ключей (включая уникальные) — до 8192.

Файлы таблиц HyTech

Физически таблица размещается в нескольких файлах:
- описатель таблицы (файл с расширением HTB), содержащий сведения о структуре записи таблицы;
- постоянная часть таблицы состоит из двух файлов:
  - область данных (файл с расширением HDT);
  - ассоциатор (файл с расширением HAS);
- область изменений (журнал изменений) также состоит из двух частей:
  - оглавление журнала (файл с расширением HDN);
  - тело журнала (файл с расширением HDR);
- сетевые таблицы, кроме того, имеют файлы сетевых замков:
  - удаленный замок;
  - локальный замок.

Такая организация файлов называется *дифференциальной*. Позднее вы увидите, как с помощью такой организации файлов решаются проблемы согласованности по чтению и транзакциям и обеспечивается высокая параллельность в БД.

В первый момент времени после создания таблица состоит из одного *описателя таблицы*. После открытия доступа к таблице для нее создаются файлы области изменений, в которые заносятся все добавляемые записи, а также сведения об изменениях и удалениях. Не надо путать журнал изменений с традиционным журналом транзакций. Поскольку журнал изменений и есть часть базы данных, то любая транзакция сначала записывается в журнал транзакций и только после второй фазы попадает в журнал изменений. По мере заполнения журнала изменений следует выполнять операцию обновления. После выполнения операции обновления область изменений переносится в постоянную часть таблицы, а журнал изменений очищается. При выполнении этой операции символьные поля записей сжимаются за счет удаления незначащих пробелов.

Таблица — совокупность сведений о предметной области, для которой создается база данных. Таблица представляет собой массив записей. *Запись* состоит из ряда полей, каждое из которых характеризуется именем, типом данных и числом байтов, отводимых для их хранения (длиной). Кроме того, в таблице содержится описание групп и подстрок.

4. Создание пользовательских приложений

HyTech_Designer является инструментальным средством, позволяющим быстро построить приложение на языках C и Pascal на основе традиционной схемы: главное меню, иерархические подменю, панели диалога, вызываемые элементами меню и подменю.

Пользуясь HyTech_Designer, вы можете быстро сформировать иерархическое меню большой вложенности, связать элементы меню с панелями диалога, сгенерировать файл ресурсов и программу, использующую файл ресурсов, оттранслировать программу выбранным вами транслятором, получить EXE-файл и, не выходя из среды HyTech, протестировать его.

HyTech_Designer генерирует как файл ресурсов, содержащий меню, панели, диалога, таблицы строк, так и программу, которая использует этот файл ресурсов. Сгенерированная программа является основой, которую вы должны дополнить своим исходным кодом.

Элементы главного меню обычно вызывают располагающиеся вертикально подменю. Элемент подменю может иметь дополнительную «горячую» клавишу, которая отображается справа от названия элемента.

HyTech_Designer, системные ресурсы и диалоговые библиотеки

HyTech_Designer является приложением, использующим диалоговые библиотеки *HyTech_Vision* новой версии. Ранее диалоговые библиотеки поставлялись в составе дистрибутива HyTech 1.6. Для языка C они назывались BxHT_DLy.LIB, где x — буква модели, а y — буква версии (E — английская версия, R — русская версия), для языка Pascal поставлялся ряд модулей (Units) HYT_???.PAS.

В новой версии поставляются одни и те же диалоговые библиотеки HyTech как для русской, так и для английской версии, а различия вынесены в файлы системных ресурсов.

Файл системных ресурсов HTSYSTEM.HRS содержит стандартные панели диалога («Загрузить файл», «Сменить каталог», «Помощь»), а также сообщения об ошибках среды и DOS.

Организация работы пользователя с клавиатурой и «мышью» аналогична методам работы в таких средах, как Borland C, Borland Turbo Pascal, пакете Borland Turbo Vision и так далее.

5. Интерфейс СУБД HyTech с языками программирования C и Pascal

В данном разделе приведено краткое описание функций СУБД HyTech 2.0, доступных из прикладных программ, написанных на языках C/C++ в среде MS-DOS для компиляторов фирм «Borland Int.» и «Microsoft Corp.» Ввиду ограничений на печатный материал описание функций для языка Pascal мы опустим.

Функции собраны в основную библиотеку для модели LARGE, пригодную для работы с любым из указанных компилятором. Кроме того, для каждого компилятора имеется дополнительная библиотека, учитывающая его особенности.

HYTECH.LIB — основная библиотека функций СУБД;
HYTECH_T.LIB — дополнительная библиотека для компилятора TurboC 2.0 фирмы «Borland Int.»;
HYTECH_B.LIB — дополнительная библиотека для компиляторов C++ фирмы «Borland Int.» (Turbo C++ 1.0, Borland C++ версий 2.0, 3.0 и 3.1);
HYTECH_M.LIB — дополнительная библиотека для компиляторов C и C++ фирмы «Microsoft Corp.» (MSC 5.0, 6.0 и 7.0);
HYTECH_Z.LIB — дополнительная библиотека для компиляторов C и C++ фирмы «Zortech».

Описание функций собрано в .H файлах:
HYTECH.H — описание типов данных и функций СУБД;
HTECOD.H — список кодов завершения и описание функций обработки ошибок.

Требования к ресурсам, необходимым для работы библиотечных функций:
- используемая часть стека составляет не более 2 килобайтов;
- для работы с extended памятью необходимо наличие XMS-драйвера версии не ниже 2.0;
- для использования expanded памяти необходимо наличие EMS-драйвера версии не ниже 4.0.

Применение дополнительной или расширенной памяти существенно ускорит работу с большими таблицами.

Для языка Pascal функции собраны в пять модулей и поставляются в виде исходных текстов, что позволяет подготовить их для соответствующей версии компилятора. Кроме того, имеются файлы, описывающие интерфейсную часть всех модулей и содержащие константы и типы.

В функциях СУБД часто используется механизм *call-back*, то есть пользовательских функций, вызываемых СУБД для выполнения необходимых действий, например, модификации тела записи. Особенности среды программирования TurboPascal налагают ряд *ограничений* на описание пользовательских функций.

Пользовательские функции не должны быть локальными, так как локальные функции имеют дополнительный «скрытый» параметр. Кроме того, СУБД вызывает пользовательские функции, применяя длинные вызовы, поэтому следует самому позаботиться о том, чтобы это были FAR функции. Для этого не забудьте о директиве {$F+} перед их описанием или установите триггер в меню Option.Compiler.Force far calls.

Поскольку в такой статье описать все функции просто невозможно, мы остановимся на самых важных и интересных.

Отметим только то, что перед вызовом функций СУБД необходимо инициализировать СУБД, а перед выходом из программы — завершить ее работу. Для работы с таблицами имеются известные функции — создать/ открыть/удалить и другие, а для работы с индексами — построить/сменить/ удалить. Для модификаций таблицы предусмотрены операции двух классов: по одной записи и пакетами — добавление/удаление/модификация.

Поисковые операции

Это наиболее важный класс операций, и на нем мы остановимся более подробно.

Все поисковые операции можно условно разделить на *пять классов.*
1. Поиски записей.
2. Поиски значений ключей.
3. Межтабличные поиски.
4. Операции над группами.
5. Прочие поиски.

К первому классу относятся поиски, результатом которых является некоторое подмножество записей в искомой таблице. Такие поиски также удобно разделить на *три группы*:
- атомарные поиски;
- логические операции над атомарными поисками;
- преобразования результатов.

Атомарным поиском назовем такой поиск записей, когда значение одного из элементов записи (поля, группы или подстроки) удовлетворяет определенному условию.

Для *логических связок* результатов атомарных поисков применяются операции отрицания, логического «И», логического «ИЛИ» и исключающего «ИЛИ».

К третьей группе относятся операции *преобразования списков* значений ключа в список записей. Например, можно получить список записей, у которых поле «год_рождения» принимает одно из следующих значений { 1950, 1960, ..., 1990 }.

Ко второму классу операций можно отнести те операции, результатом которых является список значений заданного ключа. Например, найти список кабинетов, которые посещали больные из заданной области.

Над полученными списками значений ключей также можно выполнить логические операции («И», «ИЛИ», «НЕ» и исключающее «ИЛИ»).

В третий класс операций выделены операции межтабличного поиска. Эти операции позволяют выполнить слияние нескольких таблиц для порождения виртуальных результирующих таблиц. СУБД HyTech имеет средства для преобразования виртуальных таблиц в реальные.

Операции этого класса позволяют реализовывать межтабличные отношения «один к одному», «один ко многим» и «многие ко многим». Например, пусть в таблице AREA содержится описание территорий («код_области», «область»), в таблице PEOPLE — описание заболевших людей («паспорт», «фамилия», «код_области» и так далее). Тогда для формирования

таблицы WHERE («фамилия», «область») нужно выполнить операцию слияния таблиц AREA и PEOPLE через общее поле «код_области», в результате которой будет получена новая таблица.

Операции *четвертого класса* позволяют вычислить некоторые библиотечные *функции* (MAX, MIN, SUM, AVERAGE, COUNT), разбив подмножество записей на некоторые группы (сгруппировав их по значениям нескольких полей).

К пятому классу относятся операции, которые трудно классифицировать. Например, операция вычисления гистограммы значений какого-либо ключа или получение среднего значения ключа на подмножестве записей.

Концепция поиска. «Фотография» состояния БД

Работа с БД в сети порождает ряд проблем, связанных с отслеживанием изменений данных, производимых одними пользователями в процессе выполнения поисков другими пользователями. При выполнении сложных запросов к БД пользователь начинает поиск при одном состоянии данных, а завершает в совершенно другом.

Если не предпринять специальных действий, то при достаточно динамичной работе нескольких пользователей окажется невозможным, например, подготовить отчет (выполнить ряд сложных взаимосвязанных поисков), так как через некоторое время полученные промежуточные результаты уже потеряют актуальность (устареют) и общий результат будет содержать явную бессмыслицу.

Для того чтобы разрешить возможные коллизии и создать у пользователей иллюзию их единственности, в СУБД HyTech используется механизм «фотографий» (моментальных снимков) состояния БД.

Этот механизм можно описать следующим образом. При выполнении первого атомарного запроса из всей группы поисковых операций состояние таблицы «фотографируется». Последующие операции, используя этот «снимок», выполняют поиск для того же состояния БД, которое было на момент первого запроса. Все внесенные другими пользователями изменения не видны для данного пользователя. Это позволяет завершить всю сколь угодно длинную цепочку поисковых запросов без нарушения логической целостности.

Моментальный снимок БД позволяет вам обеспечить согласованность по чтению и транзакциям и достичь максимального показателя параллелизма.

Межтабличные операции

Правильно спроектированная БД обычно состоит из некоторого количества таблиц, организованных так, чтобы избежать избыточного хранения данных.

Однако для прикладных задач наверняка потребуется построить виртуальную таблицу, содержащую данные из разных взаимосвязанных таблиц. Эту операцию можно назвать слиянием таблиц через общий ключ.

В результате слияния будет установлено соответствие записей двух исходных таблиц (образованы пары «запись-запись»).

В приведенном примере таблицы соотносятся через поле «код республики» как *один-много*.

Отношение *много-много* часто не имеет физического смысла, так как невозможно установить однозначное соответствие между записями таблиц (будут построены все возможные сочетания записей, часть из которых не несет логической нагрузки). Как известно, для таких отношений обычно используется третья таблица.

Описанное слияние можно распространить на число таблиц, большее двух. Например связь трех таблиц может иметь вид:

```
A - B - C      или      A      или      A
                       / \             / \
                      B   C           B - C.
```

Во втором примере видно, что таблицы B и C связаны между собой через таблицу A. Такую связь можно назвать *объединением*. Обычно такая связь будет представлять собой декартово произведение (множество всех возможных сочетаний) записей таблиц. Однако в случае, если связки A-B и A-C являются отношением *один-много*, то результирующие записи объединения будут иметь вид:

```
A B x
A x C
```

Здесь x означает «пустое» (NULL) значение.

«Пустое» значение вместо записи одной из таблиц означает, что в данной связке записей трех таблиц эта запись не имеет логической связи с двумя другими и поэтому не может быть определена. Например, если таблица A описывает клиентов, таблица B — заказы, а таблица C — зарплату, то объединение таблиц может дать:

ABx — сведения о заказах клиента (ничего не можем сказать о получаемой зарплате — нет связи с зарплатой);

AxC — сведения о зарплате (ничего не можем сказать о сделанных заказах — нет связи с заказами);

ABC — такая запись будет неверной, так как устанавливает несуществующую связь заказов с зарплатой.

Обратите особое внимание на то, что данная реализация операции Join в языке SQL коренным образом отличается от SQL других СУБД. Именно такой подход позволил решить известную для языка SQL проблему «декартова взрыва».

В результате слияния/объединения нескольких таблиц будет установлено соответствие *запись-запись... запись* для этих таблиц, каждый элемент которого можно рассматривать как композитную запись, состоящую их записей исходных таблиц, попавших в это соответствие.

Операции над группами

Часто бывает необходимо рассчитать значения поля для некоторого подмножества записей, разбив множество на группы. Например, подсчитать среднюю заработную плату по отделам предприятия. Это можно сделать так: найти записи для первого отдела и подсчитать для них среднее, затем для второго и так далее.

Функциями, вычисляемыми для группы, могут быть среднее значение/количество/максимум/минимум/сумма.

Кроме вычисления функций, предусмотрена возможность дополнительной фильтрации каждого элемента группы. Например, запретить расчет для определенных отделов. Для выполнения фильтрации элементов группы используется механизм call-back функций, реализуемых пользователем.

Работа с результатами поиска

Результаты, полученные после выполнения поисковых операций, сохраняются в памяти (оперативной или дополнительной) либо на диске.

Для доступа к результатам поиска используется «файловый» подход: результаты надо открыть, потом можно читать и перемещаться по ним, после завершения работы результаты надо закрыть. Ненужные результаты надо удалить, освободив тем самым занимаемые ими ресурсы.

К таким операциям относятся следующие: *открыть доступ к результатам поиска, читать результаты поиска, перемещение логического указателя по результатам поиска, пометить текущую запись результата, позиционировать результат на помеченную запись, многоключевая сортировка результатов поиска, вывод результатов поиска в таблицу HyTech.*

Управление доступом

Для администратора БД имеются функции: сменить криптоключ закрытия данных/пароль доступа на чтение/пароль доступа на запись.

Сетевая поддержка

Как известно, при работе в сети возникают проблемы, связанные с разделением доступа нескольких пользователей к таблице или ее записям. Это связано, например, с тем, что при параллельной модификации одной и той же записи второй пользователь может затереть изменения первого.

Для разрешения такого рода коллизий в СУБД используется известный механизм замков (захватов). Замки позволяют пользователю сети уведомить других пользователей о том, что он распоряжается данной записью (группой записей или таблицей) в данный момент времени. Имеются замки, действие которых распространяется на таблицу целиком, группу записей или отдельную запись таблицы.

Захват всей таблицы целиком позволяет одному из пользователей блокировать таблицу и запретить другим абонентам сети выполнять любые операции модификации (добавление, удаление или редактирование записей).

Захват одной или нескольких записей позволяет пользователю запретить другим абонентам сети выполнять операции модификации или удаления одной или нескольких захваченных записей. Они ждут освобождения замка или истечения времени тайм-аута.

Все операции, связанные с установкой замков, также ждут освобождения ресурсов или истечения тайм-аута.

Предельное время ожидания (величину тайм-аута) можно менять.

Закрытие таблицы автоматически снимает все замки с таблицы и записей, установленные пользователем, открывшим таблицу.

Замки, оставшиеся захваченными из-за «зависания» или отказа станции, освобождаются автоматически при повторном запуске СУБД на данной станции и открытии доступа к заданной таблице.

Среди функций захвата можно выделить следующие: *определить свободные от захвата записи и захватить их, захватить таблицу/запись таблицы/результаты поиска (множество записей), освободить захваченную таблицу/запись таблицы/результаты поиска, установить тайм-аут для сетевых операций.*

6. Интегрированная среда. Язык запросов QBE

Для пользователя-непрограммиста в СУБД HyTech имеется интегрированная среда с событийно-управляемым интерфейсом и развитым Browse_HyTech.

Прекрасные средства диалога позволяют пользователю легко преодолеть все барьеры между БД и пользователем. Серией щелчков «мыши» или нажатием кнопок клавиатуры можно без труда выполнить любые операции:

- создать/изменить/удалить таблицу БД,
- назначить/сменить пароли доступа к БД,
- осуществить импорт/экспорт БД (DBF или ASCII),
- выполнить любой сложности запрос к БД по образцу (QBE),
- просмотреть результаты поиска как в табличном виде, так и по заготовленным формам,
- сформировать отчет по результатам поиска,
- наиболее подготовленным пользователям предоставлена возможность формирования запросов на языке SQL,
- обработать таблицы/результаты поиска средствами деловой графики

и многое другое.

Пункт меню «Запрос» нужен для формулирования запросов к таблицам. При выборе этого пункта меню на экране появляется подменю (рис. 1).

Вопросы, которые вы задаете о содержимом таблиц, называются запросами. При составлении запроса вы можете указать:

Запрос QBE	F8
Программа SQL	
Получить количество	Alt+F9
Просмотреть результат	F9
Построить таблицу	Ctrl+F9
Новый результат	Ctrl+D
QBE в SQL	Ctrl+S
Удалить сообщения	Ctrl+O

Рис. 1

- какие таблицы содержат нужную информацию;
- какие поля вы хотите включить в запрос;
- какие записи вы хотите включить в запрос.

Вы также можете использовать запросы для поиска, изменения, добавления или удаления информации. Вы можете
- находить записи в таблице;
- удалять записи из таблицы;
- добавлять новые записи в таблицу;
- изменять содержимое полей таблицы.

Окно запроса QBE

Необходимо обратить особое внимание на то, что в СУБД HyTech, в отличие от всех других СУБД, выполнение запросов любой сложности разделено на два этапа:
- подсчет числа записей-целей и формирование списка,
- формирование итоговой таблицы (считывание результатов).

Поэтому вам предоставляется возможность перед построением итоговой таблицы узнать число записей-целей, просмотреть виртуальную итоговую таблицу, при необходимости сузить или расширить запрос и после этого строить таблицу-ответ.

Запрос в ячейке

В ячейках таблицы QBE может быть задано несколько типов запросов.
- *Поисковое условие*: значение поля записи результата будет удовлетворять заданному условию.
- *Переименование поля*: вы явно задаете имя поля выходной таблицы.
- *Вычисляемое выражение*: к выходной таблице будет добавлено поле, которое будет вычислено на основе других полей записи.
- *Групповой оператор*: к выходной таблице будет добавлено поле, показывающее минимальное/максимальное/среднее/суммарное значение исходного поля в группе или просто количество записей в группе.

Ячейку запроса можно заполнять как вручную, так и выбором значений из гистограммы (возможность быстрого получения гистограммы любого поля также является отличительной особенностью СУБД HyTech).

Первая колонка таблицы QBE предназначена дла задания типа запроса. Если в этой колонке ничего не задано, предполагается обычный поисковый запрос.

В этой колонке могут быть значения:

DELETE — запрос на удаление; при наличии в таблице других запросов выполняется первым;

INSERT — запрос на добавление; выполняется после запросов на удаление и модификацию;

FIND — просмотр группы записей в исходной таблице.

При многотабличном запросе разные таблицы связываются общими полями, которые должны содержать одинаковый вид информации. Например, таблицы AREA и SEQ имеют общие поля, в которых содержатся коды областей, названные KOD_OBL. Имена полей могут различаться, но типы этих полей должны быть совместимы. Для указания общих полей следует ввести в них элемент-пример.

В качестве элемента-примера можно использовать любой символ, нельзя использовать пробелы. Главное условие связывания таблиц — в общих полях должны находиться одинаковые символы или одинаковые наборы символов.

Связывание таблиц

Как показано на рис. 2, таблицы AREA и SEQ связаны с помощью символа «X». Допустим, нужно получить информацию об Иванове. Так как в таблице SEQ нет данных о республике, где проживает Иванов, эту информацию можно получить из таблицы AREA. Таким образом, запрос будет выглядеть, как показано на рис. 2.

В таблице результата будут присутствовать помеченные в обеих таблицах поля (рис. 3).

7. Язык запросов SQL и обработка SQL утверждений

Одним из основных понятий является курсор.

Курсор — рабочая область, располагающаяся в памяти. В курсоре располагаются скомпилиро-

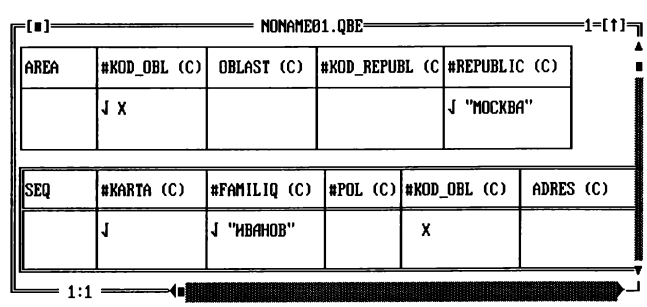


Рис. 2

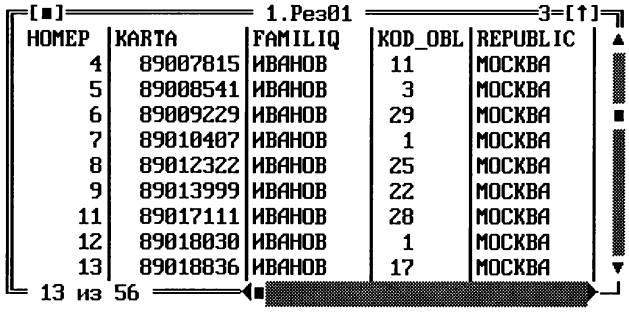


Рис. 3

ванный оператор SQL, коды завершения операций и различные системные переменные. Каждый курсор имеет имя, по которому на него можно ссылаться из операторов SQL, а также номер, по которому к курсору ссылаются при вызове функций. Обычная последовательность вызовов функций в программе:

SqlInit	Инициализация
...	
SqlOpenTable	Открыть (зарегистрировать) таблицу HyTech и указать для нее пароли доступа к данным
...	
SqlCursorOpen	Создание нового курсора (можно опустить)
...	
SqlCompile	Трансляция SQL-оператора
...	
SqlBind	Задание связи между переменными SQL и переменными программы
...	
SqlExec	Выполнение оператора
...	
SqlOpenResults	Задание буфера для результата, открытие результатов
...	
SqlReadResults	Выбор строки результата (можно вызывать несколько раз)
...	
SqlCloseResults	Закрытие доступа к результатам
...	
SqlCommit	«Закрепить» результаты операций (в настоящей версии можно опустить)
...	
SqlCursorClose	Освободить курсор
...	
SqlCloseTable	«Отпустить» таблицу HyTech
...	
SqlDone	Завершение работы

Оператор SELECT предназначен для отбора информации из одной или нескольких таблиц. Он позволяет отобрать как строки из таблиц, так и столбцы.

Формат:

```
SELECT Список_выводимых_столбцов
FROM   Список_используемых_таблиц
[ WHERE Условие_отбора ]
[ GROUP BY Список_полей_группировки
[ HAVING Условие_отбора_групп ] ]
[ ORDER BY Список_полей_для_сортировки ]
```

Список_выводимых полей задает поля, которые можно получить, считывая результаты поиска.

Список_используемых_таблиц задает таблицы, из которых выбирается информация.

Условие_отбора определяет, какие именно записи отбираются.

Список_полей_группировки определяет поля, по которым производится группировка информации.

Условие_отбора_групп определяет группы, которые войдут в выходную таблицу.

Список_полей_для_сортировки определяет порядок записей в выходной таблице.

Подзапросы

Подзапрос — запрос, который используется внутри другого SQL-предложения. Подзапросы часто применяются внутры конструкции WHERE операторов SELECT, INSERT, DELETE, UPDATE.

В данной реализации есть 2 операции с подзапросами:

```
IN и NOT IN
```

Формат конструкции IN:

```
Поле1 IN ( SELECT Поле2 FROM ... )
```

Это выражение вычисляется так:
1) находятся все записи, удовлетворяющие условию, заданному в подзапросе;
2) по полученным записям находится множество значений ПОЛЯ2 в них;
3) отбираются все записи в ГЛАВНОМ запросе, у которых ПОЛЕ1 имеет значение, принадлежащее множеству значений, полученному на шаге 2.

В подзапросе можно использовать еще один вложенный подзапрос и так далее. Количество вложенных подзапросов ограничено оперативной памятью и размером стека.

Объединение нескольких запросов

Результаты нескольких запросов можно объединить в один. Это распространяется пока только на оператор SELECT. Объединение запросов осуществляется с помощью конструкции UNION. В результирующий запрос входят все записи из исходных запросов, одинаковые записи **не отбрасываются**.

При объединении надо учитывать то, что все объединяемые запросы должны иметь одинаковое количество отбираемых полей (выражений). Типы отбираемых полей могут не совпадать. Если они не совпадают, то преобразуются к общему типу (если это возможно).

В выражениях SQL там, где может стоять константа, можно использовать переменные. Каждую такую

переменную необходимо связать с переменной базового языка C или Pascal. После этого ссылка на SQL-переменную будет означать обращение к соответствующей переменной C или Pascal.

Можно объявить и несвязанную SQL-переменную. Тогда доступа к ней из базового языка не будет.

Формат:

```
var   Список_переменных
```

Пример:

```
var   i, j, k;
```

Объявление переменной должно быть до первого ее использования. Связь SQL-переменной и переменной базового языка устанавливается функциями SqlBind или SqlArrayBind. Отменить связь можно функцией SqlUnbind. В выражениях перед переменной должен стоять знак «:».

Переменные можно использовать

- в условиях отбора записей (конструкция WHERE)

```
where table.field1 = :var1 and table.field2 between :var2
and 100;
```

- в условии отбора групп (конструкция HAVING)

```
having max(table.field)  :var1;
```

- в выражениях отбора колонок (список выражений после ключевого слова SELECT в операторах INSERT или SELECT)

```
select table.field * :var1;
```

- в выражениях для новых значений полей в операторе UPDATE

```
update ttt set field = field * :var1
```

Интерпретатор языка HL_SQL

Для удобства работы с БД, помимо интегрированной среды, вы можете воспользоваться интерпретатором языка HL_SQL, который является процедурным расширением языка SQL.

Процедурное расширение

В СУБД HyTech реализованы также:
if-оператор,
оператор выражения,
while-оператор,
составной оператор,
оператор вывода.

8. Транзакции

```
BEGIN WORK Имя_транзакции TABLE Список_таблиц начать
транзакцию по указанным таблицам.
```

Можно начинать несколько транзакций. Каждая транзакция должна производиться по своим таблицам и должна иметь уникальное имя. После начала транзакции начинается отслеживание всех изменений, происходящих в указанных таблицах. Эти изменения невидимы для других пользователей, пока не выполнится оператор 'COMMIT WORK'. Если выполнится оператор 'ROLLBACK', то все изменения в таблицах, включенных в транзакцию, потеряются (произойдет откат).

Управление транзакциями

Для обеспечения логической непротиворечивости данных (целостности) в процессе модификации таблиц, а также для повышения устойчивости к аппаратным сбоям в СУБД HyTech поддерживается механизм транзакций.

Проиллюстрируем работу этого механизма на примере. Пусть необходимо перенести некоторую сумму с одного счета на другой. Очевидно, что операция должна состоять из следующих шагов:

- снять сумму с одного счета (модификация записи 1);
- добавить сумму на другой счет (модификация записи 2).

После выполнения первого шага таблица будет находиться в противоречивом состоянии (нарушен баланс счетов на величину, снятую с первого счета). При параллельной работе нескольких пользователей это является ошибкой (другой пользователь видит таблицу в неверном состоянии). Кроме того, второй шаг может завершиться с ошибкой (например из-за захвата записи 2 другими пользователями). Аппаратный сбой в момент между выполнением первого и второго шага вообще приведет к утере контроля над целостностью таблицы, так как задача, «знающая» о снятии суммы с первого счета, завершилась аварийно.

Попытка решить задачу с использованием блокировки всей таблицы на момент выполнения операции замедлит работу всех пользователей сети и не сможет обеспечить восстановление после сбоев.

Для разрешения подобных противоречий и предназначен механизм транзакций. В данном случае перед выполнением операции СУБД уведомляется о начале транзакции на таблице счетов. СУБД возвращает задаче идентификатор начавшейся транзакции, после чего задача выполняет необходимые действия (шаги 1 и 2). В процессе их выполнения другие пользователи сети видят таблицу в первоначальном состоянии, как до выполнения шага 1. Пользователи не ограничены в операциях поиска над таблицами, задействованными в активной транзакции. Операции редактирования для таких таблиц ждут завершения транзакции. При удачном завершении операций модификации обеих записей следует выполнить команду завершения транзакции. После этого все произведенные изменения станут видимыми для всех пользователей сети, а таблица или таблицы станут доступными для редактирования.

В случае невозможности выполнения очередного шага, например из-за недоступности ресурса, транзакция должна быть завершена аварийно. В результате аварийного завершения все выполненные с начала транзакции изменения уничтожаются. Задействованные таблицы возвращаются в первоначальное состояние (откатываются). Аналогичный откат происходит и после аппаратного сбоя. Первое же обращение любого пользователя сети к таблице вызовет выполнение процедуры отката.

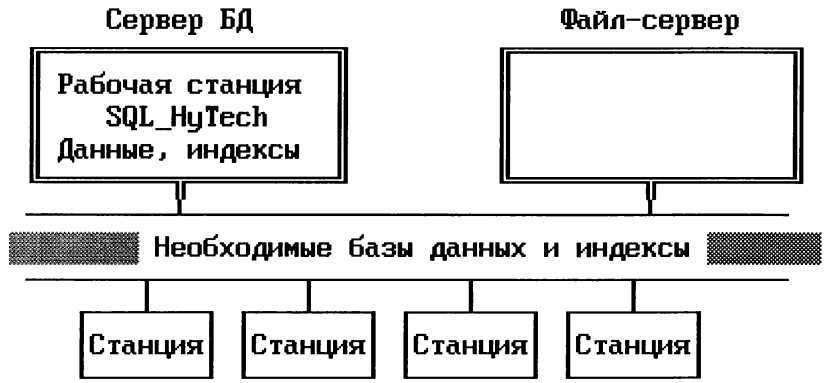


Рис. 4

Для предотвращения «зависания» установлен контроль за временем выполнения транзакций. При превышении допустимого предела транзакция помечается как подлежащая завершению вне зависимости от результатов выполнения промежуточных шагов. Пользователь, начавший транзакцию, не сможет продолжать операций с таблицей, кроме операций освобождения от захвата записей, результатов или таблиц, и должен прервать выполнение транзакции выдачей команды аварийного завершения. Другие пользователи сети видят таблицы транзакции в первоначальном состоянии и могут свободно ими распоряжаться (в том числе использовать их в транзакциях).

Для предотвращения взаимной блокировки используются известные правила:

- все требуемые блокировки запрашиваются сразу,
- после первого же неудачного захвата происходит отказ от операции.

Все действия, выполняемые от начала транзакции до ее завершения, должны производиться без неоправданных задержек (например ожидания ввода с клавиатуры), в противном случае работа пользователей в сети сильно замедляется.

9. HyTech_SQL сервер баз данных

Диспетчер отработки SQL-запросов и общие БД с индексами находятся на выделенной мощной рабочей станции (не ниже IBM PC 386, рис. 4).

Задачи пользователей формируют и посылают на эту станцию различные запросы, включая сложные поиски, транзакции, добавления, удаления, модификацию, сортировку, группирование данных и так далее.

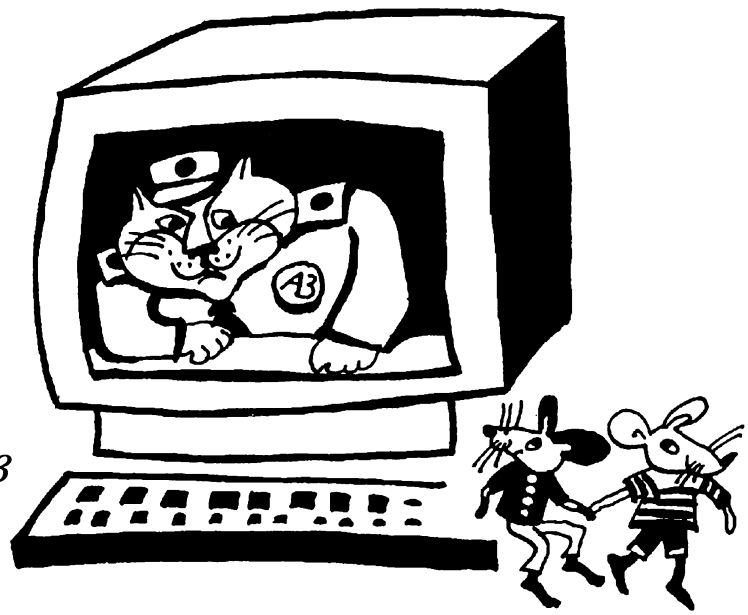
SQL_HyTech принимает эти запросы и, никуда не пересылая рабочие и промежуточные массивы, формирует ответы на запросы. При этом результатом SQL-запроса может быть как сама таблица, так и число записей-целей, чтобы можно было уточнить или расширить поисковое предписание.

Одним из возможных решений задачи обмена информацией между клиентом и сервером является использование буферных файлов, доступных для обеих программ на файл-сервере. В этом случае клиент помещает свой SQL-запрос в один из заранее оговоренных файлов, а сам ждет итоговую таблицу. Выполнив запрос, сервер отсылает итоговую таблицу прямо на файл-сервер и переходит к следующему запросу. Другим, хотя и более сложным решением этой задачи в локальных сетях может служить непосредственный обмен информацией между программами. Сервер отсылает итоговую таблицу непосредственно на станцию клиента. Стандартным решением в этом случае является использование средств фирм-производителей сетевого программного обеспечения. В случае сети NETWARE вы можете воспользоваться интерфейсными библиотеками между языками программирования высокого уровня и протоколом IPX/SPX, поставляемыми фирмой «Novell». Другие фирмы-производители поставлют аналогичные продукты для своих сетевых операционных систем. Практически все они поддерживают наиболее распространенную систему NETBIOS фирмы IBM. ■

Рекомендуемая литература

1. **Ковалевский С.С., Малярский А.Н., Зайцев К.С. СУБД HyTech — шаг к реальному масштабу времени// Мир ПК. — 1991. — № 8.**
2. **Гетманский А.Д., Топчян Б.Х. СУБД: так какая же быстрее ?// Мир ПК. — 1992. — № 2.**
3. **Ковалевский С.С., Малярский А.Н., Зайцев К.С. Взгляд на серверы баз данных// Мир ПК. — 1992. — № 4.**
4. **Rushmore в России// Computer World. — 1992. — № 48.**
5. **Нечаев В.А., Малярский А.Н. Объектные библиотеки для языков Си и Паскаль// Мир ПК. — 1993. — № 1.**
6. **Корпорация AT&T одобрила выбор СУБД HyTech для системы взаиморасчетов АО «ТелеПорт-ТП» с клиентами// Computer World. — 1993. — № 37.**
7. **Ю.В.Сальникова. Какая же СУБД быстрее// Журнал д-ра Добба. — Январь-февраль '94.**
8. **HyTech News. — 1993, 1994. — №№ 1-9.**

Прыжок Ягуара



К. Шереметьев

Компьютерная жизнь полна парадоксов. Но, пожалуй, главный ее парадокс — это использование MS-DOS на 32-разрядных компьютерах. Могучие микропроцессоры, готовые включить форсаж защищенного режима и унестись в небо, вынуждены «ползать по аэродрому» в реальном режиме, связанные устаревшими ограничениями MS-DOS. Однако для MS-DOS уже написано слишком много программ, чтобы от нее отказаться, а программировать непосредственно в защищенном режиме настолько сложно, что это по плечу немногим.

Как известно, микропроцессор 80386 в защищенном режиме фактически является нормальным процессором и предназначен для построения высокопроизводительных многозадачных и многопользовательских систем. С точки зрения программиста, 80386 состоит из двух совершенно различных устройств. При включении компьютера устанавливается реальный режим, работающий с 16-битовыми регистрами и сегментированной памятью, в которой адрес задается в виде пары <сегмент:смещение>, и сегмент записывается в сегментный регистр. В реальном режиме 80386 полностью совместим с микропроцессором 8086 и не содержит принципиальных отличий.

Защищенный режим предоставляет программисту следующие *возможности*:

* 32-битовые регистры,
* 32-битовые смещения,
* 3 варианта построения памяти (сегментированная, плоская, виртуальная),
* режим виртуального процессора 8086,
* четырехуровневую систему защиты,
* многозадачность,
* встроенные регистры отладки,
* дополнительный набор команд.

Защищенный режим получил свое название из-за принципа построения адресных областей. В этом режиме каждой задаче может быть выделена отдельная адресная область, защищенная от влияния других задач. При одновременной работе многих задач «зависание» одной из них не сказывается на остальных. Более того, если задача имеет низкий уровень привилегий, то она принципиально не сможет повлиять на выполнение остальных задач. Благодаря виртуальной памяти несколько задач могут иметь одни и те же логические адреса, причем размер адресных областей не ограничен размерами физической памяти.

Преобразование адресов в защищенном режиме происходит значительно более сложным способом, чем в реальном. При переключении в защищенный режим в сегментные регистры вместо сегментов записываются так называемые селекторы. *Селектор* — это индекс в таблице дескрипторов, содержащих описания сегментов памяти. Кроме того, при включении страничного механизма сегменты памяти могут находиться не непосредственно в памяти, а храниться на диске. Таким образом, реально адресуемая память состоит из оперативной и внешней памяти.

Для перехода в защищенный режим необходимо подготовить глобальную таблицу дескрипторов (GDT) и таблицу дескрипторов

прерываний (IDT). После этого достаточно выполнить команды:

```
mov     ax,1
lmsw    ax
```

В результате вы оказываетесь в защищенном режиме. Однако после этого вы, скорее всего, увидите, как компьютер перезапускается, так как первое же аппаратное прерывание не может быть обработано. Фактически дальше у вас два пути. Либо вы обеспечиваете все аппаратные и программные прерывания сами, либо используете для этого некоторый «расширитель» DOS. Наиболее популярный на настоящий момент — это продукт фирмы «PharLap». Но его недостаток в том, что приходится носить с собой загрузчик выполняемых модулей. Можно использовать операционные системы для защищенного режима, такие как Windows NT или OS/2, но они требуют больших ресурсов и на машинах стандартных конфигураций фактически больше их забирают, чем дают.

В этих условиях преимуществом обладают форт-системы, которые представляют собой независимые операционные среды и могут работать автономно.

В качестве такой операционной среды можно использовать пакет Форт-Ягуар, разработанный НПП «Ирбис». Это 32-разрядная форт-система, работающая в защищенном режиме, однако запускающаяся как обычная DOS-программа.

Форт-Ягуар — это система, которая обеспечивает стандартный набор возможностей форт-среды, но в защищенном режиме. Получаемые EXE-файлы включают запускающую часть ядра, которая и переводит компьютер в защищенный режим. Главное достоинство Форт-Ягуара в том, что программисту не требуется ломать голову над премудростями защищенного режима: он просто запускает пакет из MS-DOS и оказывается в чудесном плоском мире, где вся оперативная память доступна линейно по 32-битовому адресу. На машине с памятью 4 Мб Ягуар предоставляет приблизительно 3,6 Мб.

Для работы в форт-ассемблере есть четыре предопределенных селектора: текстовой видеопамяти, графической памяти, прерываний DOS и PSP программы.

Работа с пакетом Форт-Ягуар

Ягуар унаследовал интегрированную среду от ПараФорта, в котором, по заключению экспертизы АО «КУДИЦ», «...идея интегрированной среды доведена до логического совершенства» (рис. 1).

В редакторе вы видите проверяемое слово (Искать_строку), проверочное слово (T), выполняете его в окне интерпретатора и видите результат в стеке. Буфер слов используется для ускорения ввода и уменьшения орфографических ошибок.

Работа с Ягуаром происходит следующим образом. В первую очередь необходимо продумать логику программы и записать ее в естественном виде, в таком, как она записывается обычно в статьях по программированию. После этого вы устанавливаете курсор на первое слово и нажимаете Alt-F2. Появляется новый экран с пустым определением этого слова. Вы пишете определение, и процесс повторяется, пока вы не определите все слова программы. Таким образом, вся задача раскрывается сверху вниз, от более общих понятий к более конкретным.

Пользовательский интерфейс в Ягуаре рекомендуется создавать с помощью оконной библиотеки, которая позволяет делать это практически независимо от основной программы.

Затем начинается процесс отладки. Для каждого слова рекомендуется создавать временное проверочное слово, например T или TEST, и многократно проверять его с разными входными параметрами. Если предполагается, что программа в дальнейшем будет развиваться, то лучше проверочные слова оставлять в тексте, что позволит в дальнейшем моментально замечать отклонения при доработке слов.

Этот процесс чрезвычайно прост и эффективен. Если логика работы программы достаточно продумана, то фактически идет непрерывная разработка программы без отвлечения на бессмысленные траты времени типа ожидания компиляции или тщетных попыток представить в уме весь ход работы.

Для программиста, создающего сложные задачи, Форт-Ягуар может стать инструментом, обеспечивающим принципиально новый уровень эффективности программирования. ■

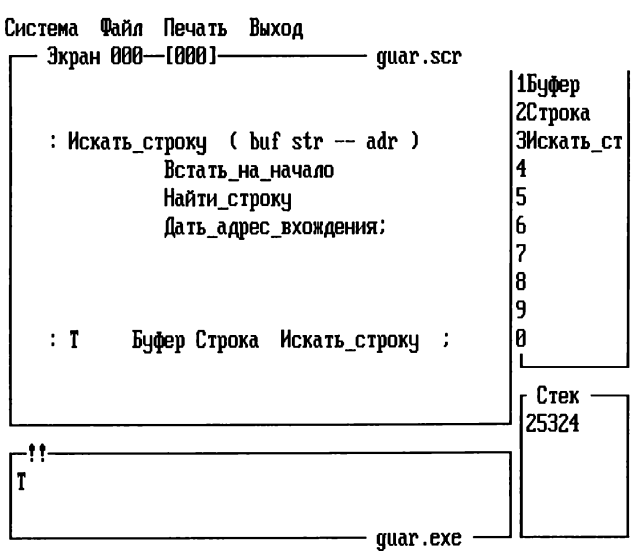


Рис. 1. Интегрированная среда

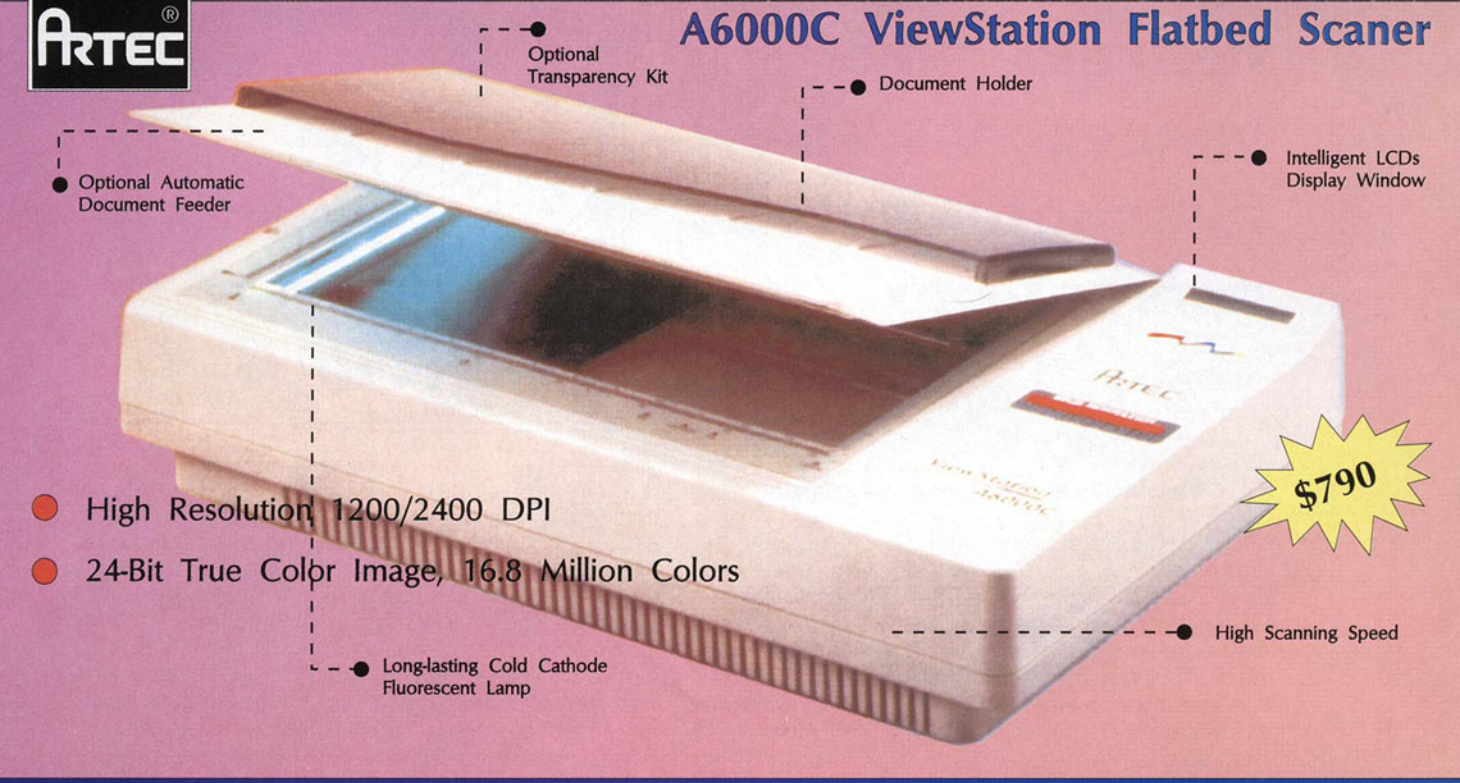
Trans-Ameritech
Enterprises, Inc.

High Performance
Computer Systems

Authorized Distributor

ARTEC®

A6000C ViewStation Flatbed Scaner



ViewScan/256

- 100-800 DPI, B/W & 256 True GrayScalel
- Magic Auto Stich/Merge Function
- PhotoStacker Plus & GO-CR Software

WalkScan/256

- 100-800 DPI
- Portable Parallel Port
- AC-adapter

ViewScan/COLOR 16M

- 100-1600 DPI
- 24-Bit Color Images, 16.8 Million Colors
- PhotoStacker Pro & GO-CR Software

BEST OF Computex BYTE

IMAGING MAGAZINE EDITORS' CHOICE

MIT

Dealers and Resellers WANTED CHP

TWAIN COMPATIBLE

Наши дилеры:
Сочи
ТОО "АРГО"
(8622) 93-6220

Московский Офис
119361 Москва
Большая Очаковская, 14

Тел. / факс : (095)
437-9745, 437-9746
430-2457, 430-9959

Наши дилеры:
Хабаровск
СП "СТЕЛЛАР"
(4212) 34-1679

Иваново	Тамбов	Екатеринбург	Челябинск	Белгород	Тула	Курган	Вятка
АО "РУСЛАН"	АО "СЕРВЕР"	ТОО "АРКОН"	АО "ПРОМАТИС"	ТОО "ЭЛТОН-Л"	"ЧЕРМЕТИНФОРМ"	ТОО "ФЕРУС"	фирма "КОРСА"
(0932) 23-3482	(0752) 33-5005	(3432) 44-1660	(3512) 65-5620	(07222) 71-552	(0872) 43-6068	(35222) 64-896	(8332) 67-5588

Смешанное программирование и подключение универсальных библиотек в Basic программах

А. Колесов

Предисловие к материалам рубрики

Сегодня мы завершаем обсуждение проблем смешанного программирования. К сожалению, ограниченный объем рубрики позволил опубликовать статью, посвященную стыковке с двоичными библиотеками, только в сокращенном варианте. Полный текст этой статьи, в которую включен дополнительный раздел о стыковке с графической библиотекой PCX Programmer's Toolkit версии 3.5 фирмы «GENUS Microprogramming», в виде отдельного файла MON_95_1.ZIP распространяется ИКЦ АПМБ в электронной версии.

Обратная связь с читателями является той основой, на которую опирается любое периодическое издание. Собственно, постоянная рубрика «MS BASIC» возникла в 1993 году, прежде всего, из ответов на довольно многочисленные вопросы Basic программистов, адресованные в информационно-консультационный центр «Ассоциация пользователей MS Basic».

Нам пишут, звонят, к нам приходят. Со многими нашими коллегами мы поддерживаем постоянную связь. Именно такие контакты помогают определить тематику статей рубрики и наиболее актуальные вопросы. Возможно, это удивительно, но за полтора года существования рубрики «MS BASIC», мы получили только одно письмо с выражением недовольства (он о приведено в этом номере). Так или иначе, публикуя некоторые фрагменты нашей переписки с читателями, мы одновременно приглашаем всех принять активное участие в работе рубрики «MS BASIC» и формировании тематики статей:

105058 Москва, а/я 4,
ИКЦ АПМБ,
А.А. Колесову,
тел. (095) 366-25-02,
E-mail: akolesov@glas.apc.org

Сколь-нибудь серьезная разработка программ неизбежно приводит к необходимости использования готовых инструментальных средств — библиотек подпрограмм, реализованных в объектном (двоичном) формате LIB/QLB. При этом надо иметь в виду, что представление библиотек в этом формате вызвано не только и не столько желанием автора скрыть исходный код, а тем, что очень часто эти библиотеки созданы совсем на другом языке программирования и могут быть подключены к Basic программе только на уровне объектного кода.

В этой проблеме можно выделить два основных вопроса.

1. Как, в принципе, подключить к программе пользователя двоичные библиотеки?
2. Как подключить внешние процедуры, написанные не в Basic формате?

Первая проблема является чисто технической, и она уже обсуждалась в нашей рубрике [3]. Второй вопрос часто возникает при работе с универсальными библиотеками, которые могут применяться с различными языками программирования (как правило, такие библиотеки пишутся на Assembler — это обеспечивает отсутствие некоторых конфликтов при смешанном программировании на языках высокого уровня).

Обычно авторы таких программных продуктов детально описывают порядок подключения библиотек к программам пользователя. И тем не менее знание основных принципов стыковки с универсальными библиотеками может оказаться нелишним. Оно может помочь и в более гибком использовании библиотек, и в переходе к другой версии языка (например от QB 4.x к PDS 7.x [1]), и в разработке собственных инструментальных средств.

Основные элементы межпроцедурного интерфейса

Суть проблемы стыковки заключается в том, что стандартные интерфейсы обращения к процедурам,

передачи аргументов и само кодирование данных во всех языках программирования (Basic, C, Fortran, Pascal) несколько отличаются друг от друга. При этом, создавая инструментальные процедуры на Assembler, авторы или берут за основу какой-то определенный интерфейс, или создают некоторый свой вариант.

В отличие от Assembler входной интерфейс со стороны языка высокого уровня является фиксированным (на Basic нельзя написать процедуру, к которой можно было бы обратиться по правилам C, и наоборот). Но при этом все они имеют явные или неявные возможности коррекции своих стандартных выходных интерфейсов (на Basic очень просто написать обращение к процедуре по правилам C).

Говоря о межпроцедурном интерфейсе, следует иметь в виду следующие его элементы.

1. *Согласование имен процедур.* Почти во всех языках идентификатор процедуры, указанный в исходном тексте программ, используется и в объектном модуле. А на языке C процедура с именем fact на самом деле подразумевает идентификатор _fact — впереди автоматически ставится знак «подчеркивание». Однако этот символ является недопустимым для описания имен процедур в Basic, поэтому нельзя обратиться к данной процедуре, просто указав CALL _FACT.
2. *Порядок передачи аргументов.* В языке C передача аргументов процедуры (запись в стек) производится справа налево, во всех остальных языках — в обратном порядке.
3. *Способ передачи аргументов* — по ссылке или значению. Внутри QB 4.x передача выполняется только по ссылке (в PDS 7.1 можно и по значению). Именно несоответствие в способе передачи является очень частой ошибкой, поэтому нужно обязательно внимательно познакомиться с описанием конкретной процедуры.
4. *Способ адресации аргументов при их передаче по ссылке.* В большинстве языков программист может выбирать разные модели памяти (наиболее распространены MEDIUM и LARGE, в первом случае данные хранятся в ближнем сегменте данных, во втором — в дальних сегментах), в соответствии с которыми выбирается тип адресации с помощью ближнего и дальнего (полного) адреса. В MS Basic фактически используется одна модель, которая представляет собой комбинацию упомянутых: динамические массивы (в PDS также и строковые переменные) хранятся в дальних сегментах, а все остальные данные (в том числе и описатели массивов и строк) — в ближнем. Соответственно, все аргументы передаются в процедуры только с помощью ближних ссылок.
5. *Способы хранения и описания данных.* Для простых числовых переменных проблем нет — их формат унифицирован. А вот более сложные структуры данных могут иметь отличия, и их надо учитывать в работе (подробнее об этом будет сказано далее).

Чтобы не отвлекаться от основной темы этой статьи, напомню читателям, что суть различных вариантов передачи параметров (пункты 2-5) уже обсуждалась ранее в [4] на примере смешанного программирования Basic-Assembler.

Как будет видно дальше, Fortran реализует универсальный алгоритм передачи аргументов, который чаще всего применяется в ассемблерных библиотеках. Основным его отличием от Basic является отсутствие каких-либо специальных описателей структур — все аргументы (простые переменные, строки, массивы) передаются одинаково, с помощью адреса самой переменной. Именно поэтому в дальнейшем некоторые вопросы стыковки процедур будут рассматриваться на примере Basic-Fortran. В свою очередь, передача аргументов в C отличается от Fortran только формированием фактического имени процедуры и последовательностью записи в стек.

Как изменить в Basic стандартное обращение к процедуре

Для изменения стандартного обращения к процедурам в Basic используются расширенные варианты операторов CALL, CALLS и DECLARE (NonBASIC Procedures). Можно применять различные комбинации этих операторов [2], [3], но универсальным является описание процедуры в явном виде, чего Basic программисты очень не любят, с помощью оператора DECLARE, который в общем виде выглядит так:

```
DECLARE {SUB:FUNCTION} ИмяПроцедуры [CDECL] _
     [ALIAS "АльтернативноеИмя"] [([СписокПараметров])]
```

(Большими буквами указаны ключевые слова, в квадратным скобках — необязательные параметры, в фигурных, через знак : — альтернативные варианты ключевых слов.)

Ключевое слово CDECL устанавливает некоторые элементы интерфейса языка C: порядок передачи параметров справа налево и преобразование имени процедуры в объектном модуле (добавляется знак подчеркивания впереди, и все буквы меняются на прописные). Например, описание

```
DECLARE SUB Fact CDECL ...
```

указывает, что в действительности в объектном модуле имя процедуры — _fact.

А параметр ALIAS позволяет вообще заменить указанное в исходном тексте программы имя внешней процедуры на любое другое. Этот способ используется, когда настоящее имя в объектном модуле нельзя воспроизвести в Basic в явном виде (например недопустимые символы или слишком длинный идентификатор):

```
DECLARE SUB Fact ALIAS "Abs^+@" ...   ' Имя в объектном
                                      'модуле: Abs^+@
```

Способ передачи аргументов и их адресация указываются в списке параметров:

```
[{BYVAL:SEG}] переменная [AS тип] [, ...] ...
```

Ключевое слово BYVAL указывает, что передача переменных (только для числовых простых переменных) выполняется по значению, а SEG — с помощью ссылки с использованием дальнего адреса (сегмент и смещение). Здесь нужно обратить внимание, что реализованная в QB 4.x возможность передачи аргументов по значению с применением круглых скобок типа

```
CALL Sub1 ((НачальноеЗначение))
```

на самом деле является лишь имитацией этого варианта передачи со стороны вызывающей программы, которую можно представить в таком виде:

```
ВременнаяПеременная = НачальноеЗначение
CALL Sub1 (ВременнаяПеременная)
```

То есть вызываемая процедура Sub1 в любом случае имеет дело с адресом переменной, в данном примере временной, изменение значения которой не влияет на вызывающую программу.

Передача строковых переменных

В отличие от простых числовых переменных, при работе с более сложными структурами данных, кроме знания адреса и типа переменной, программе требуется некоторая дополнительная информация: длина, размерность и прочее. Поэтому при работе с такими переменными в программе часто используются специальные описатели (дескрипторы), формат которых может отличаться не только в разных языках, но и в версиях одного и того же языка (например описатель строковых переменных в QB 4.x и PDS 7.x). Именно поэтому в универсальных библиотеках, как правило, используются наиболее простые способы передачи подобных сложных структур, которые довольно легко реализовать на языке высокого уровня.

Здесь необходимо отметить, что именно использование дескрипторов (разделение данных и их описаний) позволяет реализовать идею динамических данных, свободно перемещаемых по памяти, изменяющих свои размеры. Как будет показано далее, представить динамические структуры в виде статических — это простая задача, а наоборот — довольно проблематичная. Поэтому смешанное программирование, например

```
Basic (вызывающая программа) -> C, Fortran, Assembler
(вызываемая)
```

реализуется без проблем по всем типам данных и структур, а в обратном направлении — с существенными осложнениями.

Например строковая переменная представляет собой одномерную последовательность байтов, число которых должно каким-то образом определяться в программе. Известны два принципиальных решения подобной проблемы: либо задать длину последовательности в явном виде, например

```
<N — кол-во элементов> <элемент 1> ... <элемент N>,
```

либо указать конец последовательности с помощью некоторого специального значения элемента, который не должен использоваться в реальных данных, например

```
<элемент 1> ... <элемент N> <Специальный элемент — конец данных>
```

В Basic традиционно используется первый вариант. Раньше число байтов хранилось в самой строке (нулевом байте), поэтому ее длина была ограничена 255. В QB ближний адрес переменной и описание длины переместились в специальный дескриптор, в PDS 7.x этот дескриптор изменился, адрес стал дальним. Короче говоря, дескриптор время от времени меняется, и это создает проблемы стыковки с процедурами, которые настроены на конкретный тип дескриптора.

Типичный пример — привязка ассемблерных процедур PDS 7.x к новому дескриптору сделала невозможным их использование в QB 4.x. Конечно, было гораздо проще сделать эти библиотеки пригодными для любой версии языка, но, очевидно, фирма «Microsoft» решила в данном случае создать пользователям больше стимулов к покупке PDS 7.x. (Хотя для опытного программиста это легко решаемая проблема, нужно только написать вспомогательную процедуру на Assembler в десяток строк.)

В языках C и Fortran используется второй вариант, в качестве ограничителя строки — нулевой байт. Именно такой способ наиболее часто используется в универсальных интерфейсах передачи строк байтов, в том числе, и при обращениях к системным функциям ROM-BIOS, DOS. Например, в описании функции DOS 3dH («открыть файл») указано: «DS:DX — адрес строки с именем файла».

Для правильного формирования имени файла и заполнения регистров в программе можно сделать такие операции:

```
'NameFile$ — переменная с именем файла
  NameNew$ = NameFile$ + CHR$(0) 'строка в формате C, Fortran
  InpReg.DX = SADD(NameNew$)
    InpReg.DS = VARSEG(NameNew$)        ' для QB 4.x
    ' для PDS 7.x и VB/DOS предыдущую строку нужно
    ' заменить на InpReg.DS = SSEG(NameNew$)
```

А можно использовать другой вариант, переписав сначала содержимое строковой переменной в целочисленный массив (процедура SymbolConv, листинг 1):

```
REDIM NameNew%(1) 'фиктивное резервирование массива
CALL SymbolConv(NameNew%(),NameFile$) 'перезапись в массив
       InpReg.DX = VARPTR(NameNew%(1))
       InpReg.DS = VARSEG(NameNew%(1))
```

И хотя второй вариант выглядит несколько более тяжеловесно, именно на него я прошу обратить внимание читателей: во-первых, он уже не зависит от версии MS Basic (я работаю и с QB, и с PDS), а во-вторых, он нам пригодится в дальнейшем.

Передача массивов

В отличие от строковых переменных, описание массива требует большего числа параметров. Например, дескриптор массива, используемый в Basic, включает не только полный адрес памяти, но и число индексов, а также значения нижней и верхней границы для каждого индекса.

В языках Fortran и C массив передается точно так же, как и простая переменная, с помощью адреса области данных. Предполагается, что вызываемая процедура должна знать все об этом массиве сама (типичная статическая организация данных) и сформировать его дескриптор с помощью его описания в явном виде, например

```
C ==== Fortran ====
           SUBROUTINE Proc3(Array)
           DIMENSION Array(15,30)
```

При этом считается, что если в вызывающей программе этот массив был определен как DIMENSION Array(20), то это личная проблема программиста, и компилятор ничем ему не поможет.

Как же передать адрес массива из Basic в процедуру, реализованную в стиле Fortran? Для этого рассмотрим такие варианты:

```
' ==== Basic ====
REDIM Array (1 TO 15, 1 TO 30)
'динамический массив
CALL Proc1 (Array())
CALL Proc2 (Array(1,1))
CALLS Proc3 (Array(1,1)) '= равносильно — CALL
                         ' Proc3 (SEG Array(1,1))
```

Первый вариант однозначно соответствует передаче массива по правилам Basic: передается ближний адрес дескриптора массива, с которым Fortran программа работать не умеет.

Во втором случае передается ближний адрес простой переменной. Этот вариант был бы работоспособным, если бы массив Array в Basic был описан как статический (находится в ближнем сегменте данных), а Fortran программа была откомпилирована с моделью памяти MEDIUM.

Но в нашем случае массив определен как динамический, поэтому Basic программа передает не адрес самого элемента массива (он характеризуется дальним адресом), а адрес временной простой переменной (она формируется в ближнем сегменте данных), которая фактически передается в процедуру, то есть эта конструкция для динамического массива на самом деле представляет такую операцию:

```
' ==== Basic ====
Tmp = Array(1,1): CALL Proc2(Tmp): Array(1,1) = Tmp
```

Надо четко представлять этот алгоритм передачи в Basic элементов динамического массива, так как он может иногда привести к некоторым неожиданным для программиста эффектам. Например в данном случае

```
' ==== Basic ====
REDIM a(1): CALL SubTest(a(),a(1)): PRINT a(1);a(2)
  ...
SUB SubTest(b(),p)
p=1: FOR I=1 TO 10: b(I)=0: NEXT
END SUB
```

будет распечатано: 1 0.

Если же массив a() будет определен как статический, то результат будет таким: при работе в среде интерпретатора — «1 0» (передается временная переменная, а не элемент массива), а при работе EXE модуля — «0 0» (передается сам элемент массива).

Третий вариант является решением требуемой задачи — в процедуру передается именно полный адрес первого по порядку элемента динамического массива.

В качестве некоторого резюме по данным примерам можно сказать следующее:

- второй вариант обращения может использоваться только при передаче из Basic данных, записанных в ближнем сегменте памяти, при этом модель памяти в Fortran должна быть MEDIUM;
- при использовании в Fortran модели памяти LARGE должен применяться вариант 3; только этот вариант годится для передачи данных, записанных в дальних сегментах, в частности, динамических массивов (для PDS 7.x также строковых переменных).

И, заканчивая рассмотрение вопроса передачи массивов, следует заметить, что по умолчанию нумерация индекса в Basic начинается с нуля, а в Fortran — с единицы. Отдельно нужно также упомянуть о правильности записи данных в многомерных массивах. Сейчас, кажется, во всех языках используется последовательность записи отдельных элементов по столбцам, хотя в Basic имеется специальная опция компилятора, меняющая этот порядок на запись по строкам.

Передача адреса строковой переменной

Прием-передача строковой переменной в Fortran (C) выполняется точно так же, как и простых числовых переменных или массивов — с помощью ссылки. В частности, такие конструкции в Fortran программе являются правильными, с точки зрения передачи адреса некоторой области памяти:

```
C ==== Fortran ====
      CALL Text('Передача строковой переменной')
         ...
        SUBROUTINE Text(strng)
C   вариант 1 — прием как строковой переменной:
        character strng
C   вариант 2 — прием как целочисленного массива:
        integer*2 strng(10)
C   вариант 3 — прием как простой целочисленной переменной:
        integer*2 strng
```

Для правильной передачи из Basic строковой переменной (ее адреса) по правилам Fortran можно использовать такие два варианта.

1. Переписать содержимое строки в целочисленный массив, а затем передать его адрес в процедуру (пригодилась процедура SymbolConv !):

```
' ==== Basic ====
Word$ = "Передача строковой переменной"
REDIM Strng%(1)     'фиктивное резервирование массива
CALL SymbolConv(NameFile$, Strng%()) 'перезапись в массив
CALLS Text(Strng%(1)) 'передача текста
```

2. Указать непосредственно адрес содержимого строки с помощью передачи его сегмента и смещения по значению (этот прием обсуждался в [4]):

```
' ==== Basic ====
Segment% = VARSEG(Word$)        ' QB 4.x; для PDS
7.x   = SSEG(Word$)
Offset% =   SADD(Word$))
CALL Text (BYVAL VARSEG(Word$), BYVAL SADD(Word$))
```

Графическая библиотека Enhanced Graphics Toolkit

Библиотека разрабатывалась в 1986-88 гг. фирмой «Connell Scientific Graphics», автор Connell (судя по всему, он же основатель и, кажется, единственный сотрудник фирмы). В настоящее время эта библиотека распространяется на правах FREEWARE. В момент появления первой версии (1986) она была создана для нового тогда монитора EGA, в последующем в нее также были добавлены режимы VGA (всего 12 графических,

начиная с EGA 640×350), а название сохранилось — сокращенно EGA Toolkit.

По сути дела библиотека EGA Toolkit является автономным вариантом графических процедур, аналогичных составу графических операторов MS Basic. Именно поэтому в свое время она привлекла внимание в первую очередь тех, кто работает с языками, не имеющими встроенных графических средств (C, Fortran). Но очень скоро возможности ее использования оценили и многие Basic программисты, которые вроде бы и имели уже «все под рукой».

Прежде всего надо отметить, что EGA Toolkit имеет существенно более широкий набор функций по сравнению с MS Basic (около 60 в версии 1988 г.). Более того, она позволяет реализовать видеоэффекты, получить которые встроенными средствами Basic, в принципе, нельзя — управление логикой рисования, наплывающие видеостраницы, создание и перемещение видеостраницы 800×600 (и это в режиме экрана 640×350!) и многое другое — для этого лучше посмотреть демонстрационную программу библиотеки.

Как серьезный плюс я рассматриваю и то, что она реализована только для работы в физических координатах, а это означает, что используются только целочисленные аргументы, нет затрат времени и памяти на преобразование типов данных. (Любой приверженец работы в логических координатах может написать свою процедуру в десяток операторов для выполнения нужных ему преобразований.)

Говоря о EGA Toolkit, полезно вспомнить дискуссию на тему «что лучше — встроенные или автономные средства». Ведь динамичное развитие этой библиотеки: расширение видеорежимов, функций (почти в три раза за полтора года!), оперативное устранение отдельных дефектов и прочее — было возможно только за счет ее автономности, не говоря уже о том, что вся библиотека распространялась в исходных текстах, то есть любой программист мог расширять ее самостоятельно. Не менее важным мне представляется то, что эту библиотеку можно... не использовать вообще, заменив ее на что-то более подходящее.

Подключение библиотеки

Однако с точки зрения нашей сегодняшней темы главным в библиотеке является возможность ее применения в различных языках. Как этого добился автор?

Основной модуль библиотеки реализован на языке Assembler — EGA16C (в первой версии было всего 16 функций). Кроме него, есть еще несколько автономных модулей с дополнительными функциями, которые можно подключать по мере необходимости. Обращение ко всем 45 функциям основного модуля выполняется через одну входную точку, так же как через один вход идет обращение к огромному числу функций BIOS и DOS, и имеет на Basic следующий вид:

```
DIM Regim%(1 TO 2), Parameters%(1 TO 10)
  ...
CALLS EGA16C(Regim%(1),Parameters%(1))
'Regim%(1)  — код функции
'Regim%(2)  — число параметров (для переменного
            ' числа параметров)
'Parameters%()  — список параметров
            ' (в зависимости от функции)
```

Как видно из этого примера, обращение к входной точке библиотеки производится по правилам Fortran: все аргументы записываются в два целочисленных массива, каждый из которых, в свою очередь, передается с помощью полного адреса. Аналогично производится обращение к функциям других модулей библиотеки: через одну входную точку в каждом модуле.

Основные ассемблерные модули представлены в виде двоичной библиотеки, хотя для желающих есть и их исходные тексты. Для удобства работы с функциями библиотеки, по крайней мере, на этапе ее освоения, в состав пакета входят модули-переходники для Basic, C и Fortran — HLEV(.BAS, .C, .FOR):

```
ПрограммаПользователя -> HLEV.BAS -> EGA16C
```

В них каждой функции соответствует отдельная процедура со своим смысловым именем и конкретным составом аргументов. В листинге 2 приведен фрагмент трех процедур модуля HLEV.BAS. Здесь я бы обратил внимание на следующие моменты:

1) если массив состоит из одного элемента, то передается простая переменная или даже константа;
2) при передаче строковой переменной (TEXT) используется самый простой вариант — ее длина передается в виде отдельного параметра Func(2).

Для использования в среде MS Basic библиотеки EGA Toolkit нужно выполнить такие действия (на примере QB 4.5):

1) создать библиотеку EGA.LIB:

```
LIB EGA.LIB+EGA16C.OBJ [+ другие объектные модули
библиотеки]
```

2) создать Quick библиотеку:

```
LINK /q EGA.QLB,EGA.LIB,,BQLB45.LIB
```

3) загрузить QB вместе с библиотекой:

```
QB.EXE /L EGA
```

На практике, скорее всего, вы будете пользоваться довольно ограниченным числом функций, для этого проще обращаться к ним напрямую — CALL EGA16C(...). Но если вам удобнее работать с модулем HLEV.BAS, то его нужно загрузить в QB командой <File:Load> или создать расширенную библиотеку (HLEVEGA = HLEV+EGA):

1) QB /l EGA HLEV.BAS,

далее создать командой <Run:Make Library> библиотеку с именем HLEVEGA и работать с ней:

2) QB /l HLEVEGA.

Обратите внимание, что при переходе от QB к PDS не возникает никаких проблем в связи с изменением структуры дескриптора строковых переменных: либо вы работаете, прямо обращаясь CALLS EGA16C (никакие дескрипторы не используются), либо все проблемы преобразования и передачи данных решает модуль HLEV.BAS в среде конкретной версии языка.

В заключение напоминаем читателям, что полный текст этой статьи в виде файла MON_95_1.ZIP распространяется ИКЦ АПМБ в электронном виде. В него включен дополнительный раздел, посвященный стыковке с графической библиотекой PCX Programmer's Toolkit

версии 3.5 фирмы «GENUS Microprogramming», где приведена краткая характеристика пакета, описание структуры его библиотек (они имеют принципиально другую организацию по сравнению с EGA Toolkit) и даны рекомендации по решению некоторых проблем, связанных с переходом от QB 4.x к PDS 7.x. ■

Литература

1. **В. Артюхов. Как обмануть коварство «Microsoft» // Монитор. — 1994. — № 6.**
2. **А. Колесов, О. Павлова. Для тех, кто не хочет расставаться с Фортраном// Мир ПК. — 1992. — № 1.**
3. **А. Колесов, О. Павлова. Профессиональная работа в среде Microsoft QuickBASIC // Монитор. — 1993. — № 2, 3.**
4. **А. Колесов. Смешанное программирование: Basic-Assembler // Монитор. — 1994. — № 7, 8.**

Уважаемая редакция,

с интересом прочитал работу г. А. Колесова «О выборе оптимальных программных конструкций» для языков семейства MS Basic («Монитор» № 5, 1994 г.). Если ограничиться средствами собственно языка (без несколько трудоемкого процесса использования ассемблерных вставок), то приведенный в работе пример формирования строковой переменной известной длины можно радикальным образом ускорить. Для этого следует использовать описание языка и операторы VARPTR, PEEK, POKE. Например, так (дополним листинг 1 из указанной работы вариантом 3):

```
'--------------- Вариант 3 ---------------
DIM LinePtr AS LONG
  Start! = TIMER
  FOR c% = 1 TO Cicle%
    a$ = SPACE$(nf%)
    LinePtr = VARPTR(a$)  ' смещение заголовка
                          ' строковой переменной;
  'смещение первого символа строковой переменной:
    LinePtr = 256*PEEK(LinePtr+3)+PEEK(LinePtr+2)
    FOR n% = 1 TO nf%
       POKE LinePtr, n%: LinePtr = LinePtr + 1
  NEXT: NEXT
  TimeT! = TIMER - Start!
  PRINT «Вариант 3: «; TimeT! - TimeCicle!
```

Прогон листинга 1 (самостоятельный exe-файл) из работы г. Колесова А., дополненного вариантом 3, дал следующие результаты (для качественной оценки тип компьютера не имеет особого значения).

Вариант 0: 0.43
Вариант 1: 37.41
Вариант 2: 17.03
Вариант 3: 1.83

И, очевидно, это еще не предел. Пользуясь данным приемом, следует иметь в виду, что определять адрес при помощи функции VARPTR лучше в непосредственной близости от использующих его операторов PEEK/POKE, поскольку динамический объект может незаметно для пользователя сменить свое местоположение. Полезно самому время от времени дефрагментировать область памяти строковых переменных при помощи функции FRE("").

С уважением, Мошков А.В.

Уважаемый г-н Мошков!

Большое спасибо за ваше письмо. Предложенный вами вариант прекрасно подтверждает оба тезиса моей статьи:

1) используя оптимальные конструкции, можно существенно повысить эффективность программ;
2) для повышения быстродействия следует минимизировать операции создания динамических объектов (в данном случае была исключена функция CHR$).

Признавая элегантность предложенного вами решения, тем не менее, я хотел бы обратить ваше внимание на следущие моменты.

1. Использование переменной типа LONG не гарантирует вас от проблем с формированием адреса смещения в операторе

```
LinePtr& = 256 * PEEK(LinePtr&+3) + PEEK(LinePtr&+2)
```

Ведь операции в правой части выполняются по правилам переменной INTEGER и лишь затем преобразуются в переменную LONG, то есть при смещении больше 32 кб будет ошибка. Эту проблему легко решить, заменив константу 256 (INTEGER) -> 256& (LONG). Кстати, этот простой пример очень хорошо показывает известный недостаток MS Basic — отсутствие беззнаковых целочисленных переменных.

2. Для определения адреса смещения первого символа строковой переменной совсем не обязательно обращаться к структуре ее описателя, в MS Basic для этого есть специальный оператор: LinePtr&=SADD(a$).
3. Использование оператора PEEK/POKE предполагает устанавливать адреса текущего сегмента данных оператором DEF SEG. В Вашей конструкции предполагается, что он указывает на ближний сегмент данных, но это вовсе не обязательно. Поэтому более точно при определении адреса смещения еще и произвести установку:

```
DEG SEG (для QB 4.x) или DEF SEG = SSEG(a$)  (для PDS 7.x)
```

В своей статье я сознательно не рассматривал вариант с использованием операторов PEEK/POKE, чтобы не отвлекаться на некоторые проблемы их применения (вы их очень правильно описали). Именно поэтому я привел в статье довольно тривиальный пример заполнения строки байтами 1-255.

В реальной жизни гораздо чаще встречаются более сложные варианты, например перезапись численного массива в строку. Именно такой случай с использованием пяти разных вариантов подобного преобразования данных, в том числе и ассемблерных процедур, рассматривается в моей статье «Работа с двоичными файлами в MS Basic» (она включена в наш электронный журнал «BasicArt», выпуск 4, 1995 г.). При этом, используя PEEK/POKE, приходится постоянно манипулировать адресом текущего сегмента, что приводит к снижению быстродействия. И в результате скорость вариантов, аналогичных моему второму и вашему третьему, становится примерно одинаковой.

В заключение я еще раз благодарю за ваш интересный и полезный комментарий и приглашаю вас к дальнейшему участию в обсуждении проблем программирования в среде MS Basic.

А. Колесов

Главному редактору журнала «Монитор»
т. Д.Э.Бройтману

Уважаемый главный редактор.

Обращаюсь к вам в общепринятой форме, хотя в первом слове я сильно сомневаюсь после прочтения 6'94. Статьи в журнале якобы для программистов-практиков подходят скорее для журнала юных «программистов». Не буду голословным. Вот яркий пример из 6'94. Статья некого «эксперта» А. Колесова на стр. 71, 2-я колонка, 3-й абзац. Фраза «... обращение к системной функции DOS с номером 16h» !!!!

Это просто смешно — перепутать функцию DOS и BIOS. Дальше можно просто не читать — А. Колесов может быть экспертом в каких угодно областях, но только не в программировании.

Системный программист НВП «Эскорт»
Фомичев А.В.

Уважаемый г-н Фомичев!

Вы совершенно правы — прерывание процессора с номером 16h относится к группе функций ROM-BIOS (Read Only Memory Basic Input/Ouput System, обычно используется просто термин BIOS). Хотя, честно говоря, большинству прикладных программистов (а именно для них была написана статья) совершенно все равно, к какой группе функций относится то или иное прерывание. Поэтому многие авторы сознательно, чтобы не отвлекать внимание читателей, употребляют термин «системные функции» — ведь обращение к функциям BIOS и DOS производится совершенно одинаково. Часто по этой же причине для простоты объяснения BIOS рассматривается как составная часть DOS (хотя это, конечно, не совсем точно).

Лично я не вижу ничего страшного в такой «вольности», так как любой программист (для которого это важно) знает, что системное прерывание 16h — это обращение к одной из функций BIOS, а 21h — к функциям DOS. Тем не менее, я еще раз признаю — виноват, исправлюсь. И хотя мне очень жаль, что из-за моей оплошности досталось и уважаемому (без всяких сомнений с моей стороны) главному редактору Д.Э. Бройтману, все-таки после прочтения вашего письма у меня осталось чувство искреннего удовлетворения, ведь большинство людей просто не читает статьи, которые им кажутся неинтересными, а вы прочли ее до конца. И самое главное, если указанный вами недостаток является самым слабым местом в статье, то... лично у меня есть повод быть довольным ее качеством.

Кроме того, я хотел бы вам задать вопрос, ответ на который я нигде не могу найти уже в течение многих лет, — кто такой системный программист? Это человек, который разрабатывает любые системы или именно операционные системы? А может, это тот, кто умеет грамотно работать с чужими системами или хотя бы устанавливать их на жесткий диск? В частности, попадает ли в эту категорию Билл Гейтс, не только основатель и президент фирмы «Microsoft», но и создатель систем Basic нового поколения? Сам он себя называет Basic программистом.

С наилучшими пожеланиями,
А. Колесов

Листинг 1

```
'''''''''''''''''''''''''''
'  Модуль SYMCONV.BAS     '
'''''''''''''''''''''''''''
SUB SymbolConv (Text$, Integ%())
'
'    Переписать содержимое строковой переменной
'    в массив + добавить CHR$(0) в конец
''''''''''''''''''''''''''''''''''''''''''''''''''''''

    temp$ = Text$ + STRING$(2, 0)
    LenS% = LEN(temp$) \ 2
    REDIM Integ%(1 TO LenS%)
    FOR n% = 1 TO LenS%
      Integ%(n%) = CVI(MID$(temp$, n% * 2 - 1, 2))
    NEXT n%
END SUB
```

Листинг 2

```
'*********************************************
'*        Фрагмент модуля HLEV.BAS           *
'*          графической библиотеки           *
'*        Enhanced Graphics Toolkit          *
'*-------------------------------------------*
'*  Стыковка с основным модулем библиотеки   *
'*      CALLS EGA16C(Func(1), Param(1))      *
'*********************************************
DEFINT A-Z
DIM SHARED Param(20), Func(2)
SUB DRWLOG (Logic) STATIC
'    Установка логики графического вывода
'    (нет аналога в MS Basic)
'''''''''''''''''''''''''''''''''''''''''

    CALLS EGAC16(4, Logic)
END SUB
SUB RECTAB (ix1, iy1, ix2, iy2, fill) STATIC
'    Вывод на экран прямоугольника
'    fill = True  - закраска прямоугольника
' Basic аналог:
'   LINE (ix1, iy1)-(ix2, iy2),,  {B:BF}
''''''''''''''''''''''''''''''''''''''''''

     Param(0) = ix1: Param(1) = iy1: Param(2) = ix2:
Param(3) = iy2
     IF (fill) THEN Fun = 11 ELSE Fun = 10
     CALLS EGAC16(Fun, Param(1))
END SUB
SUB TEXT (Word$) STATIC
'   Вывод текста в графическом режиме
'   (не путать с оператором PRINT!)
'   1) позиционирование в графических координатах
'   2) управление логикой рисования
''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'    Func(0) = 17: Func(1) = LEN(Word$)
    temp$ = tx$ + "null": inc = 1
    FOR n = 0 TO (Func(1) / 2)
      Param(n) = ASC(MID$(temp$, inc + 1, 1))
      Param(n) = (256 * Param(n)) + ASC(MID$(temp$, inc,1))
      inc = inc + 2
    NEXT n: CALLS EGAC16(Func(1), Param(1))
END SUB
```

UNIX™ EXPO

THE OPEN SYSTEMS & NETWORKING SHOW

1995 FEBRUARY 7-10

The first Open Systems exhibition . . .

. . . for those in Banking & Industry

Unix Expo '95 будет проходить в Москве с 7 по 10 февраля 1995 года в павильоне Форум на Красной Пресне.
Ежедневно с 10.00 до 18.00, 10 февраля с 10.00 до 16.00

ПРОПУСК (НА ОДНОГО ЧЕЛОВЕКА)

Для получения бесплатного пропуска на выставку, а также бесплатного каталога представленных на выставке компаний, заполните билет и представьте его при входе.

Принимаются полностью заполненные ксероксные копии пропуска

Организация _____

Имя _____

Должность _____

Почтовый Адрес _____

_____

_____

Номер Телефона _____

Номер Факса _____

Род Занятий _____

Лица до 18 – ти лет, не входящие в состав групп из университетов и технических учереждений, допущены на выставку не будут. На этой выставке представлена новая компьютерная технология, расчитанная на специалистов и представителей деловых кругов, а не на широкого потребителя. Допущены будут только лица, имеющие визитную карточку своего предприятия.

Закажите пропуск сегодня!

И Вам не надо будет стоять в очереди перед входом в павильон!

Ддя получения пропуска на выставку и в конференц - зал пожалуйста пришлите о себе информацию по факсу в Нью-Йорк 212 750 8568 или в Москве 095 338 5942 или по электронной почте 75143,3205 или 75143.3205@COMPUSERVE.COM

Sponsored by:

BLENHEIM

Blenheim Group PLC
The largest Information Technology exhibition organizer in the world.

Organized & Managed by:

WP INTERNATIONAL, INC.

For further information on UNIX EXPO '95/MOSCOW contact WP International, Inc. in New York at (212) 702-4830 or fax (212) 750-8568

Если в CMOS испортились параметры винчестера

В. Рабец

ТЕХНОЛОГИИ

Иногда случается, что в CMOS портятся параметры жесткого диска, и тут-то пользователь, кусая локти от собственной беспечности, обнаруживает, что у него нет ни Rescue дискеты, созданной с помощью Norton Utilities, ни документации к винчестеру, ни бумажки с записью параметров диска, ни отвертки со справочником. Что делать?

Запускать SysInfo или NDiags бесполезно — вы узнаете те же неправильные параметры, которые сидят в CMOS.

Хорошо, если диск — из числа стандартных для BIOS типов: во-первых, можно просто тупо перебрать, загружаясь после каждого выбора, во-вторых — прикинуть параметры с помощью изложенных далее методик и выбрать по ним диск, учитывая возможную погрешность числа цилиндров. Ситуация с User type, когда число головок, цилиндров и секторов нужно вводить самому, чуть-чуть похуже.

В любом случае богатую пищу для размышлений дает Partition table. Например, запускаем Disk Editor из Norton Utilities 6.0, выбираем в меню Physical disks, так как логического C: нет или он неправильный, — Object/Drive/ Physical disks. Выбираем Hard disk 1 и смотрим его сектор 1, цилиндр 0, головку 0 (Object/ Physical sector/...), настраиваем программу просмотра на Partition table (View/as Partition table). Видим такую, например, таблицу деления диска.

Есть некоторая вероятность, что последний раздел кончается в последнем секторе последней дорожки последней поверхности. В рассматриваемом примере, как и во многих других случаях, это не так, тем не менее, глядя в таблицу и учитывая, что нумерация цилиндров и головок начинается с нуля, а секторов — с единицы, можно предположить, что на диске 1023 цилиндра, 9 головок и 26 секторов на дорожку. Как правило, анализ Partition table позволяет точно определить число секторов и головок, а вот с цилиндрами остается небольшая неопределенность — так, например, Extended раздел после разметки программой DOS FDisk кончается не на последнем, а на предпоследнем цилиндре, то есть приведенная таблица соответствует на самом деле диску с 1024 цилиндрами); из-за неграмотной разметки или по другим причинам часть диска может не использоваться и так далее. Но даже с указанным в CMOS неполным числом цилиндров машина наверняка загрузится и будет работать. Уточнить количество цилиндров поможет изложенная далее методика.

Устанавливаем в SetUp число головок, секторов и цилиндров, заведомо большее реального, чтобы иметь возможность использовать такие значения в Disk Editor. Задав в Disk Editor цилиндр 0 и сектор 1, читаем головку 0, 1, 2 и так далее, пока не получим сообщение «Error on Hard Disk 1. Sector not found» для головки, которой нет. Так определяется число головок (обычно порядка десяти). Аналогично, установив цилиндр 0 и головку 0, читаем секторы 17, 18 и т.д. и определяем число секторов на дорожку (наиболее вероятные значения — 17, 26-29, 35-41). Наконец по уже известным параметрам и емкости диска оцениваем число цилиндров и, установив головку 0 и сектор 1, читаем цилиндры. Не упускайте из виду возможное наличие дефектных секторов — каждую ошибку чтения для надежности лучше продублировать, изменив неисследуемые параметры.

Таблица

System	Boot	Starting Location			Ending Location			Relative Sectors	Number of Sectors
		Side	Cylinder	Sector	Side	Cylinder	Sector		
DOS-16	Yes	1	0	1	8	279	26	26	65494
EXTEND	No	0	280	1	8	1022	1	65520	173862
unused	No	0	0	0	0	0	0	0	0
unused	No	0	0	0	0	0	0	0	0

К сожалению, некоторые типы дисков нормально работают при практически любом сочетании установленных параметров, и ошибка чтения возникает, только когда сквозной номер выбранного сектора выходит за пределы диска. На таких дисках данный метод применим лишь для уточнения числа цилиндров при уже известных количествах секторов и головок. Бывают диски, которые при неправильной установке вообще не читаются.

И, наконец, на самом последнем этапе, заполняя таблицу диска в CMOS, не забудьте, что нумерация секторов начинается с 1, а головок и цилиндров — с 0; в CMOS записывается число головок (секторов, цилиндров), а не номер последней головки (цилиндра).

В листинге приведена простейшая программа #H#C#S.pas (читается «номер головки, номер цилиндра, номер сектора»), выполняющая описанные операции автоматически для дисков с размерами в пределах 16 головок, 1024 цилиндра, 64 сектора на дорожку.

Сначала она читает Partition table и извлекает наибольшие из встречающихся там номеров головок, цилиндров и секторов, затем

- устанавливает максимальные параметры диска, так как чтение за пределами установленных размеров вызывает ошибку;
- сканирует головки, цилиндры и секторы до ошибки чтения;
- восстанавливает исходные параметры диска.

Поскольку при чтении могут встретиться плохие секторы, то для уменьшения вероятности влияния таких помех на результат каждый параметр сканируется дважды при чтении соседних головок или цилиндров.

Для работы программы в CMOS должен быть установлен какой-либо тип диска, так как «Disk not installed» характеризует отсутствие доступа к нему.

Программа выдает на экран не число головок и цилиндров, а их номера.

Для получения справки по запуску программы выполните #H#C#S /?

Если параметры, полученные из Partition table и путем чтения диска, не совпадают, не считая несовпадения числа цилиндров на несколько единиц, то необходимы следующие действия.

Попробуйте изменить в CMOS число головок.

Если полученное путем чтения диска число головок и секторов всегда совпадает с установленным в CMOS, то этим данным совершенно нельзя доверять, тогда следует опираться на анализ Partition table.

Если же полученное путем чтения число головок и секторов не зависит от установок CMOS, причем головок меньше 16 и секторов меньше 64, то эти данные можно считать достоверными.

Что касается цилиндров, то за исключением случая, когда оба метода дают одинаковое их число, уверенно сказать, что #H#C#S определила число цилиндров с точностью до единицы, нельзя.

К восстановленным параметрам винчестера всегда следует относиться предельно критически, ибо диск неисчерпаем так же, как и электрон. Например, #H#C#S может прочитать не только цилиндры с данными, но и цилиндр с картой дефектов, и цилиндр для парковки. Несомненно, безошибочное восстановление параметров говорит о недюжинном интеллекте пользователя. Но здесь уместно вспомнить, что умный — это тот, кто с успехом выйдет из любой сложной ситуации, а мудрый — это тот, кто в нее не попадет. ■

Листинг

```
{$A+,B-,D+,E-,F-,G-,I-,L+,N-,O-,P-,Q+,R+,S+,T+,V+,X-,Y+}
{$M 2048,65536,65536}
{ #H#C#S, (C) V.S.Rabets, 1994. Edition 09.07.94 }
uses DOS;
type tPartEntry = record Boot     : byte;
                         BegHead  : byte;
                         BegSecCyl: word;
                         Sys      : byte;
                         EndHead  : byte;
                         EndSecCyl: word;
                         OtherInfo: array [1..2*4] of
byte;
                end;
     tPartitionTable = record BootCode: array [1..$1BE] of byte;
                              PE: array [1..4] of tPartEntry;
                       end;
var PT: ^tPartitionTable absolute HeapOrg;
const Drive: byte = $80;                    { Исследуемый диск }
      MaxHeadP: byte = 0;{Max значения Head, Cyl и Sec из
Partition table}
      MaxCylP : word = 0;
      MaxSecP : byte = 1;
      ParamInt: byte = $41;       {Вектор, указывающий на таблицу }
                                  {параметров диска               }
var ParamTable: record CylCnt : word;  { Таблица параметров диска }
                       HeadCnt: byte;
                       SomeInfo: array [3..$D] of byte;
                       SecPerTrack: byte;
                       OtherInfo: byte;
                end;
var  SaveParamAddr,     { Адрес исходной таблицы параметров диска }
     SaveExitProc: pointer;
     R: registers;
     Start,
     i,w: word;
                                       { Установка параметров диска }
function SetDriveParams (ParamAddr: pointer): boolean;
                                              {ParamAddr -  @таблицы}
begin
  SetIntVec (ParamInt, ParamAddr);    { Установка int $41 или $46 }
  with R do begin
       DL:=Drive;
       AH:=9;                 { Установка таблицы параметров диска }
       intr ($13, R);
       SetDriveParams:=(Flags and fCarry)=0;
  end;
end;
function SetBigDisk: boolean;   { Установка max значений Head,
Cyl, Sect }
begin
  move ( (pointer(MemL[0:ParamInt*4]))^, ParamTable,
SizeOf(ParamTable) );
  with ParamTable do begin           { Копирование и модификация
таблицы }
       CylCnt :=1024;
       if HeadCnt<16 then HeadCnt:=16;
       SecPerTrack:=64;
  end;
  SetBigDisk:=SetDriveParams(@ParamTable);
                                        {Установка новых параметров
диска}
end;
function ReadSector (Head:byte; Cyl:word; Sec:byte): boolean;
begin
  with R do begin
       ES:=seg(HeapOrg^);                       { Размер heap задан
64кб }
       BX:=ofs(HeapOrg^);
       DL:=Drive;
       DH:=Head;
       CH:=lo(Cyl);
       CL:=Sec + (hi(Cyl) shl 6);
       AL:=1;                        { Читать 1 сектор }
       AH:=2;                          { Читать сектор }
       intr ($13, R);
       ReadSector:=(Flags and fCarry)=0;
  end;
```

Листинг (продолжение)

```
end;
function MaxWord (W1, W2, W3: word): word;      { Max из трех чисел }
var Max: word;
begin
   Max:=W1; if W2>Max then Max:=W2; if W3>Max then Max:=W3;
MaxWord:=Max;
end;
procedure ExitProcedure; far;
begin
  ExitProc:=SaveExitProc;
  if not SetDriveParams (SaveParamAddr)
     then writeln (#10'Restore disk parameters error'#7);
end;
{ ============================ MAIN ============================== }
begin
 writeln (#10'#H#C#S, ver. 1.00. (C) V.S.Rabets, 1994'#10);
 writeln ('The program try to determine number of Heads, Cylinders
         and Sectors_per_track');
 writeln ('on hard disk. It is useful in the case of CMOS trouble.');
 writeln ('The program analyzes Partition table, then it reads some
         sectors and displays');
 writeln (' the last readable Head, Cylinder and Sector_on_track.');
 writeln ('Syntax:');
 writeln ('       #H#C#S /? -  to get Help screen without work');
 writeln ('       #H#C#S    -  to explore 1st hard disk');
 writeln ('       #H#C#S /2 -  to explore 2nd hard disk');
 if ParamStr(1)=''   then                                    else
 if ParamStr(1)='/?' then halt                               else
 if ParamStr(1)='/2' then begin Drive:=$81; ParamInt:=$46 end else
     begin writeln (#10'Invalid parameters'#7); halt end;
 for w:=1 to 80 do write ('=');
 write ('Hard disk #',Drive-($80-1),':    ');
  { -------------------- Чтение раздела: ------------------------ }
  if not ReadSector (0,0,1) {Head=0,Cyl=0,Sec=1}
     then writeln ('Error reading Partition table'#7)
     else begin
       for i:=1 to 4 do with PT^.PE[i] do begin
             MaxHeadP:=MaxWord (MaxHeadP,BegHead,EndHead);
             MaxCylP :=MaxWord (MaxCylP,
                                hi(BegSecCyl)+(lo(BegSecCyl) shr6)*256,
                                hi(EndSecCyl)+(lo(EndSecCyl) shr6)*256 );
          MaxSecP :=MaxWord (MaxSecP, lo(BegSecCyl) and $3F,
                                      lo(EndSecCyl) and $3F);
       end;
       writeln ('Max values from Partition table: Head-', MaxHeadP,
                ', Cyl-', MaxCylP, ', Sector-', MaxSecP);
    end;
  { -------------- Установка новых параметров диска: -------------- }
  GetIntVec (ParamInt, SaveParamAddr); {Сохранение адреса исходной таблицы}
  SaveExitProc:=ExitProc;  ExitProc:=@ExitProcedure;
  if not SetBigDisk then begin {Установка max значений Head, Cyl, Sect}
     writeln (#10'Change disk parameters error'#7); halt end;
  { -------------------- Чтение диска: ---------------------------- }
  Start:=0; writeln;       { Поиск последней головки }
  for i:=0 to 1 do
  for w:=Start to $FF do
    if ReadSector(w,i,1) then begin    { Head=w, Cyl=0:1, Sec=1 }
       Start:=w;
       write (#13'Read Cylinder',i,',Sector 1:last readable Head -',w);
    end else begin writeln; break end;
  Start:=0; writeln;       { Поиск последнего цилиндра }
  while ( Start<1024-10 ) and ( ReadSector(0,Start+10,1) ) do begin
    inc (Start,10);
    write (#13'Read Head 0, Sector 1: last readable Cylinder -  ',Start);
  end;
  for i:=0 to 1 do
  for w:=Start to 1023 do
    if ReadSector(i,w,1) then begin    { Head=0:1, Cyl=w, Sec=1 }
       Start:=w;
       write (#13'Read Head ',i,', Sector 1: last readable Cylinder -  ',w);
    end else begin writeln; break end;
  Start:=1; writeln;       { Поиск последнего сектора }
  for i:=0 to 1 do
  for w:=Start to 64 do
    if ReadSector(i,0,w) then begin    { Head=0:1, Cyl=0, Sec=w }
       Start:=w;
       write (#13'Read Head ',i,', Cylinder 0: last readable Sector - ',w);
    end else begin writeln; break end;
end.
```

■

LOGITECH

Компьютерная периферия производства Logitech

Манипуляторы «мышь» и трекболы
удобные, точные и надежные манипуляторы для правой, левой и большой руки для любых интерфейсов

Ручные сканеры
цветные (24 бита) и черно-белые (8 битов) ручные сканеры разрешением 400 dpi; комплектуются программами распознавания символов

Цифровые фотоаппараты
автоматические портативные электронные фотокамеры для компьютера

Звуковые приставки и платы
удобные и высококачественные стерео- и монофонические устройства ввода и вывода звука через компьютер, совместимые с CD и MIDI

AudioManSoundManFotoManScanManMouseManTrackMan

Официальный дистрибьютер Logitech в России

тел. (095)934-7378
433-3457
факс (095)934-1234

Fitec

Достойные Средства для серьезной Работы

ЭЛЕКТРОННЫЕ КЛЮЧИ
SOFTWARE SECURITY, INC (США)

- Уникальный код в каждом ключе
- Встроенная энергонезависимая память
- Подключение к параллельному порту
- Прозрачность для принтера

AEGIS - обеспечивает надежную защиту готовых программ в среде DOS/WINDOWS. Позволяет шифровать файлы данных и внешних оверлеев.

ACTIVATOR - инструмент для реализации собственных идей по защите программ. Дополнительно поставляются библиотеки объектных процедур **OBJECT MODULES** и **TOOLBOX**.

DEMOKIT - демонстрационный комплект программно-аппаратных средств позволяет испытать все наши комплексы.

Более десяти лет во всем мире и уже три года в России и СНГ мы помогаем защищать авторские права разработчиков программного обеспечения.

SOFTWARE SECURITY BELARUS
Республика Беларусь, 220026
г. Минск, а/я 12
Тел./факс: (0172)45-21-03, 45-31-61,
e-mail: lev@ssb.by.glas.aps.org

НОВЕЙШАЯ КОМПЬЮТЕРНАЯ ТЕХНИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

СИСТЕМНАЯ ИНТЕГРАЦИЯ

- Комплексные решения в области создания корпоративных информационных систем
- Интеграция имеющихся информационных ресурсов в единую систему
- Разработка и моделирование сетевых проектов в нашей лаборатории.
- Оперативная техническая и информационная поддержка
- Надежное гарантийное и послегарантийное обслуживание
- Обучение персонала и консультирование
- Прсдваритсльнос апробирование решений

ЛОКАЛЬНЫЕ И КОРПОРАТИВНЫЕ КОМПЬЮТЕРНЫЕ СЕТИ

- Оборудование Ethernet 10/100, Token Ring, FDDI, ATM
- Многофункциональные модульные концентраторы
- Мосты и мульти-протокольные маршрутизаторы
- Телекоммуникационное оборудование
- Диагностическое оборудование: кабельные сканеры, рефлектометры, протокольные анализаторы
- Средства администрирования, мониторинга и управления на базе SNMP

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ОТКАЗОУСТОЙЧИВЫЕ СЕРВЕРЫ

Серверы семейства ProLiant фирмы COMPAQ:

- Модульные процессорные карты 486/Pentium с архитектурой SMP позволяют создавать наращиваемые 2-х/4-х процессорные комплексы с симметричной обработкой
- Оперативная память до 512 МБ с корректирующими кодами
- Автономная система диагностики и контроля в процессе работы со встроен ным SNMP-агентом
- Compaq TriFlcx с пропускной способностью 267 МБ/сек

МНОГОУРОВНЕВЫЕ СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ

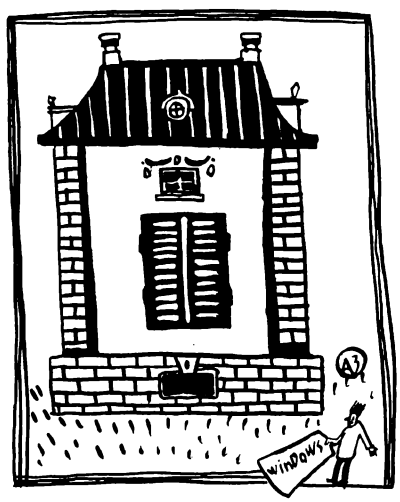
- Дисковые массивы RAID-5
- Магнитооптические накопители и роботизированные библиотеки
- Стриммеры QIC, DAT, DLT, 8mm Helican Scan
- Ленточные библиотеки
- CD-ROM: Readers, CD-Recorders, Changers, Premastering Software
- Интегрированные системы архивации и миграции данных в среде NetWare, Unix, Windows NT

COMPAQ
Авторизованный системный партнер

CROC
INCORPORATED

Россия, 103050 Москва, ул. Ермоловой 22, стр. 1.
Тел. (095) 200-1696, 299-4307
Факс: (095) 299-4625

Создание программной оболочки в среде Windows



В. Плотников

ПРОГРАММЫ

В процессе создания программных комплексов возникает необходимость объединения различных задач с целью автоматизации управления их функционированием. В различных операционных системах для этих целей используются специальные управляющие программы-оболочки, которые позволяют пользователю принимать решения по запуску отдельных подзадач данной программной системы. Причем это осуществляется не набором в командной строке команды «выполнить программу», а выбором определенного элемента меню оболочки или активизацией элемента списка дочерних процессов.

Среда Windows дает возможность создавать групповые окна в окне Program Manager. Они являются простейшими управляющими программами, объединяющими задачи по определенным признакам. Их элементами служат пиктограммы приложений Windows. Создание и модификация таких окон пользователем подробно описаны в [1]. Недостатками их использования в качестве управляющей оболочки комплекса программ являются

- отсутствие возможностей регистрации и разграничения прав доступа пользователей;
- отсутствие средств формирования блоков данных, передаваемых дочерним процессам;
- сложность организации межпрограммного обмена информацией (у групповых окон нет средств для создания таких процессов, контроля и управления ими).

Рассмотрим, как с помощью программных средств Windows 3.1 создать оболочку программного комплекса, соответствующую следующим требованиям:

1. Возможность разграничения доступа к ее приложениям.
2. Возможность запуска любых доступных в ее среде подзадач, а также контроля за ходом их выполнения.
3. Наличие средств межпрограммного обмена информацией и автоматического формирования блоков данных, передаваемых порождаемым задачам.
4. Система должна быть открытой, что подразумевает свободное добавление новых задач, удаление существующих без перекомпиляции программных компонентов оболочки.

Перейдем теперь к созданию простейшей (модельной) управляющей программы, отвечающей перечисленным требованиям. Она в исходном состоянии в среде Windows должна быть представлена пиктограммой. При попытке пользователя раскрыть ее основное окно на экране терминала должно появиться вспомогательное окно с запросом пароля у пользователя. Если оператор зарегистрирован данной оболочкой, то основное окно отображается. В противном случае, если имеем дело с нелегальным пользователем, окно будет постоянно иметь вид пиктограммы. Подзадачи этой среды в исходном состоянии должны также быть отображены в виде пиктограмм. Их окна рекомендуется сделать дочерними по отношению к главному окну программы-оболочки. Тогда закрытие основного окна позволит спрятать окна подзадач, а последующая его активизация отобразит все окна в том состоянии, в котором они были оставлены.

Передача данных порождаемым процессам и обмен информацией между ними в системе Win-

dows, как правило, основывается на технологии динамического обмена (Dynamic Data Exchange — DDE). Но он, по мнению автора, может усложнить процесс разработки программ оболочки и дочерних процессов. Необходимо отслеживать дополнительные сообщения и обрабатывать их, а доработка существующих программ с целью их подключения к оболочке выльется в написание новых программ. Гораздо проще осуществить обмен через общий блок оперативной памяти, адрес которого известен и управляющей программе, и ее дочерним задачам. Поэтому надо поблагодарить разработчиков из фирмы «Microsoft» за то, что они отвели модулям системы Windows и ее приложениям одну общую локальную таблицу дескрипторов (LDT), а также дали прикладным программам один уровень привилегий. Это позволит одним программам создавать глобальные области обмена, а другим, зная их адреса, заносить или читать оттуда информацию. При этом не возникнет прерываний (аварийных ситуаций) по нарушению прав доступа к памяти. Через данный участок памяти управляющая задача может передавать большие массивы данных дочерним, здесь могут находиться указатели на дополнительные блоки памяти, создаваемые порождаемыми процессами для обмена между собой, а также поля семафорных битов, регулирующих совместное выполнение нескольких задач. Указатель на общий блок памяти может быть передан дочерней программе через поле параметров в функции ShellExecute, ее описание читатель может найти в [2].

ПРОГРАММЫ

Система Windows обладает всеми средствами для разработки «открытых» управляющих программ, использующих средства порождения задач и организации их межпрограммного взаимодействия.

Наличие поддержки частных файлов конфигурации позволяет в среде Windows сделать нашу управляющую программу открытой системой. Файл конфигурации состоит из разделов, уникальные имена которых заключены в квадратные скобки, в следующих строках раздела располагается информация в виде пары значений, разделенных знаком «=» — имени ключа и текстовой информации, задающей его значение. В ключевых полях такого файла можно указать путь доступа к исполняемому файлу подзадачи оболочки и его имя. Добавление (удаление) дочерних процессов, поддерживаемых управляющей программой, будет сводиться к коррекции полей частного файла конфигурации. Более подробно об этих файлах можно прочитать в [3].

Иллюстрацией этих положений может служить программа-оболочка, порождающая два дочерних процесса, один из которых под именем «Приложение 1» (листинг 2) служит для занесения данных в общую область памяти, а другой — «Приложение 2» (листинг 3) — для отображения текстовой информации из этой области в своем окне.

Тексты функций управляющей программы показаны в листинге 1. Процедура WinMain предназначена для создания основного окна программы. Оно создается с кнопками минимизации и максимизации размеров окна, с изменяемыми границами окна.

Процедура WndProcShell_p является функцией основного окна программы. Рассмотрим подробно ее действия по обработке поступающих от Windows сообщений. При создании окна (WM_CREATE) удаляется пункт системного меню «Закрыть» и добавляется пункт «О программе» для отображения панели About. Попытка распахнуть окно, представленное пиктограммой, порождает сообщения WM_QUERYOPEN и WM_SHOWWINDOW. Если пользователь был идентифицирован, окно раскрывается. В противном случае — вызывается процедура создания окна запроса пароля CrWndParole. Первое раскрытие окна приводит к выделению общего блока памяти, подготовке блока параметров для дочерних программ (дескриптор основного окна оболочки, дальний указатель на общий блок глобальной памяти). Поскольку эти данные передаются в виде текстовой строки, то для исключения потери информации при наличии нулевого байта (в дескрипторе или указателе) в бок параметров вводятся дополнительно два байта: признак шифрации нулевого байта и его код-заменитель. На основе информации из файла конфигурации задачи SHELL_P.INI (пример 1) осуществляется запуск подзадач. Ключ Size раздела SizeAppl указывает на число дочерних процессов. Каждой подзадаче соответствует определенный раздел, имя которого формируется из слова «Application» и порядкового номера. Ключ NameApplication указывает на файл исполняемого образа задачи, а ключ DirApplication указывает на рабочую директорию для подзадачи. Последним сообщением, обрабатываемым оконной функцией, является WM_SYSCOMMAND. Когда оно приходит, проверяется факт выбора пользователем пункта меню «О программе», и если он имел место, то происходит отображение панели About.

Пример 1. Частный файл конфигурации задачи

```
[Size]
SizeAppl=2

[Application1]
NameApplication=appl_1.exe
DirApplication=d:\new_pl

[Application2]
NameApplication=appl_2.exe
DirApplication=d:\new_pl
```

Функция CrWndParole создает окно с полем редактирования для ввода пароля и кнопкой подтверждения (рис. 1). Процедура WndProcShell_p_1 является оконной функцией ввода идентифицирующих данных. После ввода пароля проверяется его совпадение с ключевым словом «PLOT». Правильное задание приводит к раскрытию основного окна оболочки. При несовпадении пароля появляется информационное окно, изображенное на рис. 2. Пользователь, выбирая одну из кнопок, принимает решение о дальнейших действиях с программой.

Тексты процедур, составляющих подзадачи, приведены в листингах 2 и 3. Первое приложение состоит из двух функций: WinMain и функции окна WndProcAppl_1. В процедуре WinMain происходит извлечение и расшифровка дескриптора родительского окна и указателя на область памяти из командной строки. Затем создается окно программы. Оно является дочерним и содержит поле ввода информации и кнопку подтверждения ввода (рис. 3).

Второе приложение включает в себя процедуры WinMain и WndProcAppl_2. Главная процедура аналогична предыдущему приложению, за исключением того, что здесь создается обычное дочернее окно, в котором будет отображаться информация из общей области. При попытке распахнуть окно приложения происходит проверка на наличие данных в общем блоке памяти. Если

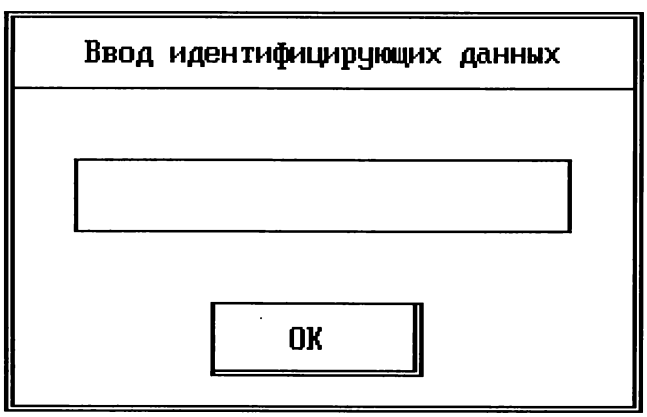


Рис. 1. Окно ввода пароля пользователя

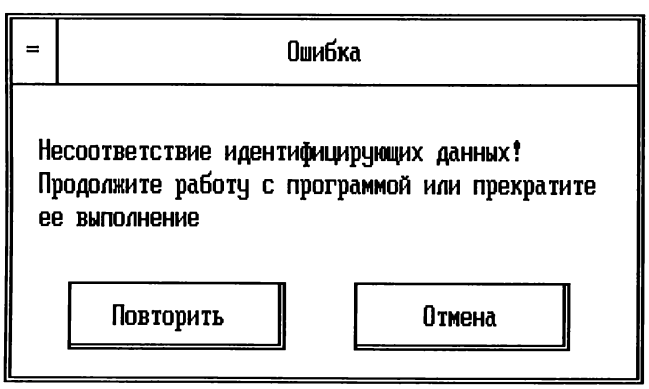


Рис. 2. Диалоговое окно ошибки при указании пароля

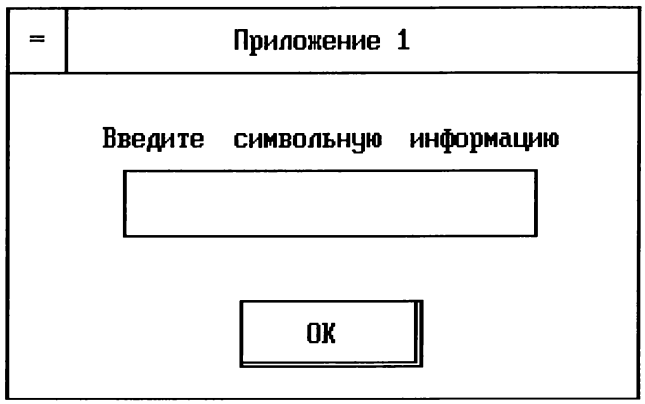


Рис. 3. Основное окно подзадачи «Приложение 1»

они отсутствуют, то высвечивается обычное диалоговое окно ошибки. Поскольку в Windows мультизадачностью управляют события, то активизация окна этого приложения для оперативного отображения данных, полученных из общей области, происходит при получении оконной функцией сообщений об изменении местоположения окна на экране или нажатии левой клавиши «мыши» при нахождении ее курсора над областью окна (WM_MOVE, WM_LBUTTONDOWN, WM_WINDOWPOSCHANGED, WM_WINDOWPOSCHANGING). Результатом реакции оконной функции на эти события является порождение сообщения WM_PAINT для обновления всей области окна.

Тексты файлов определения модулей задачи-оболочки и ее двух подзадач приведены в листинге 4. Файл конфигурации данной задачи необходимо перед ее запуском поместить в директорию с компонентами системы Windows.

Рассмотренный прием создания управляющей программы и ее задач позволяет разрабатывать компоненты, которые создаются как обычные независимые приложения Windows с перекрывающимися типами основных окон. После их отладки тип основного окна заменяется на дочерний, при этом надо учесть совместимость флагов стилей (типов) окна — [4]. Затем добавляется блок извлечения параметров из командной строки, передаваемой дочернему приложению, и корректируется частный файл конфигурации. ■

Литература

1. **Фойц С. Windows 3.1 для пользователя/ Пер. с нем. — Киев: Торгово-издательское бюро BHV, 1993.**
2. **Брюзгин А.А. Программный интерфейс для Windows 3.1. — М.: Малип, 1994.**
3. **Нортон П., Йао П. Программирование на Borland C++ в среде Windows: В 2 т. — Киев: Диалектика, 1993.**
4. **Фролов А.В., Фролов Г.В. Операционная система Microsoft Windows 3.1 для программиста. — М.: ДИАЛОГ-МИФИ, 1994.**

Листинг 1. Текст управляющей программы

```
#include        <stdlib.h>
#include        <windows.h>
#include        <shellapi.h>
#define         ID_ABOUT        1
#define         ID_ED_1         11
#define         ID_BT_1         12
#define         LN_PAROL        12
long    FAR     PASCAL WndProcShell_p
                (HWND, unsigned, WORD, LONG);
long FAR PASCAL WndProcShell_p_1
                (HWND, unsigned, WORD, LONG);
HMENU   hMenu;
int     i;
HANDLE  hInstance_, hMem;
BYTE    byFlag, byFlag1;
WORD    wId;
LPSTR   lpAdrMem;
HWND    hWndSt;
BYTE    byInst[9];
/*-------------------------------------------------------
    Массив данных, передаваемых порождаемым задачам:
 1-й элемент, если не равен 1, указывает на
необходимость замены в командной строке символов,
равных второму элементу данного массива, на нулевой
байт;
 2-й элемент определяет символ-заменитель нулевого
байта в параметрах командной строки;
 3-й и 4-й элементы массива определяют дескриптор
родительского окна (главного окна задачи-оболочки);
 Элементы с 5-го по 8-й определяют дальний указатель
на общий блок памяти, используемый для межпрограммного
обмена информацией;
 9-й элемент всегда должен быть равен нулю.
-------------------------------------------------------*/
char    cText[150], cWinText[50];
HDC     hDC;
int     iHs, i, n, l;
HWND    hWnd_1, hWndEd_1, hWndBt_1;
TEXTMETRIC      tm;
WNDCLASS        WndClass;
int     PASCAL  WinMain(HANDLE hInstance,
HANDLE  hPrevinstance, LPSTR lpCmdLine, int nCmdShow)
{
HWND    hWnd;
MSG     message;
        if (!hPrevinstance)
                                        {
byFlag, byFlag1=0; hInstance_=hInstance;
//              Создание основного окна
 WndClass.lpszClassName="SHELL_P";
 WndClass.hInstance=hInstance;
 WndClass.lpfnWndProc=(FARPROC) WndProcShell_p;
 WndClass.style=NULL;
 WndClass.hbrBackground=GetStockObject(WHITE_BRUSH);
 WndClass.hCursor=NULL;
 WndClass.hIcon=LoadIcon(NULL, IDI_APPLICATION);
 WndClass.lpszMenuName=NULL; WndClass.cbClsExtra=NULL;
 WndClass.cbWndExtra=NULL; RegisterClass(&WndClass);
 OemToAnsi("Оболочка для запуска программ", cText);
 hWnd=CreateWindow("SHELL_P", cText, WS_OVERLAPPED |
 WS_SYSMENU | WS_MINIMIZEBOX | WS_MAXIMIZEBOX |
 WS_THICKFRAME,
 CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
 CW_USEDEFAULT, NULL, NULL, hInstance, NULL);
 hDC=GetDC(hWnd); GetTextMetrics(hDC, &tm);
 ReleaseDC(hWnd, hDC); iHs=tm.tmExternalLeading+
 tm.tmInternalLeading+tm.tmHeight;
 ShowWindow(hWnd, SW_SHOWMINIMIZED);
                                        UpdateWindow(hWnd);
while (GetMessage(&message,NULL,0,0))
        {
TranslateMessage(&message); DispatchMessage(&message);
        }
return  message.wParam;
                                                }
        return  NULL;
}
long    FAR     PASCAL WndProcShell_p(
HWND hWnd, unsigned message, WORD wParam, long lParam)
////////////////////////////////////////////////////
// Функция окна программной оболочки
////////////////////////////////////////////////////
{
void    CrWndParole(HWND);
WORD    wSizeAppl;
char    cNameApplication[13], cDirApplication[125];
char    cStringApplication[14]="Application1";
        hWndSt=hWnd;
        switch (message)
 {
                case WM_CREATE:
{
        hMenu=GetSystemMenu(hWnd, FALSE);
   DeleteMenu(hMenu, 5, MF_BYPOSITION); // "Закрыть"
   DeleteMenu(hMenu, 5, MF_BYPOSITION); // "__________"
   InsertMenu(hMenu, -1, MF_SEPARATOR, NULL, NULL);
   OemToAnsi("О программе ...", cText);
   InsertMenu(hMenu, -1, MF_ENABLED | MF_STRING,
   ID_ABOUT, cText);   return NULL;
}
                case WM_QUERYOPEN:
{
  if ( byFlag == 0 ) {
  CrWndParole(hWnd); return NULL;
                     } else break;
}
                case WM_SHOWWINDOW:
{
if ( byFlag == 0 ) {// Пользователь не идентифицирован
   if(wParam != 0) ShowWindow(hWnd, SW_SHOWMINIMIZED);
   CrWndParole(hWnd); return NULL;
                   } else break;
}
                case WM_SIZE:
{
   if ( wParam == SIZEFULLSCREEN || wParam == SIZENORMAL)
    {
// Проверка факта запуска дочерних задач
                        if (byFlag1 == 0) {
// Выделение общего блока глобальной памяти
 hMem=GlobalAlloc(GMEM_MOVEABLE, 100);
 lpAdrMem=GlobalLock(hMem);
// Подготовим блок параметров для запуска дочерних
//                    программ
asm {
        mov     word ptr byInst[0],0FF01h
        mov     bx,hWnd
        mov     word ptr byInst[2],bx
        mov     bx, word ptr lpAdrMem
        mov     word ptr byInst[4],bx
        mov     bx, word ptr lpAdrMem+2
        mov     word ptr byInst[6],bx
        mov     byte ptr byInst[8],0
    }
                    for ( n=2; n < 8; n++)    {
  if ( byInst[n] == 0 ) {
                byInst[0]++;
        for (i=1; i < 256; i++) {
```

```
                                        byInst[1]=i;
                    for (l=2; l < 8; l++) {
          if ( byInst[l] == i)   {
                        byInst[1]=0; break;
                                }
                                        }
            if (byInst[1] != 0)  break;
                                }
                    break;
                        }
                                                }
            for (n=2; n < 8; n++)
            if ( byInst[n] == 0 ) byInst[n]=byInst[1];
                                        byFlag1=1;
// Чтение частного файла конфигурации оболочки
// и порождение дочерних задач
 wSizeAppl=GetPrivateProfileInt("Size", "SizeAppl", 0,
 "shell_p.ini");
                            if(wSizeAppl > 0) {
 for (i=1; i <= wSizeAppl; i++) {
     itoa(i, &cStringApplication[11], 10);
   GetPrivateProfileString(cStringApplication,
   "NameApplication", "", cNameApplication, 13,
   "shell_p.ini");
   GetPrivateProfileString(cStringApplication,
   "DirApplication", "", cDirApplication, 125,
   "shell_p.ini");
    ShellExecute(hWnd, NULL, cNameApplication,
    byInst, cDirApplication, SW_SHOWMINIMIZED);
                                }             }
                                        }
    ArrangeIconicWindows(hWnd); return NULL;
   } else break;
}
                case WM_SYSCOMMAND:
{
    if(wParam == ID_ABOUT) {
 OemToAnsi("О программе", cWinText);
 OemToAnsi("Программа-оболочка для запуска задач\n\n"
           "                (C)  Плотников В.И.  1994",
 cText);
 MessageBox(hWnd, cText, cWinText, MB_OK);
 return NULL;
                            } else break;
}
 }
return(DefWindowProc(hWnd, message, wParam, lParam));
}
long    FAR     PASCAL WndProcShell_p_1(
HWND hWnd, unsigned message, WORD wParam, long lParam)
////////////////////////////////////////////////////
//   Функция окна ввода данных, идентифицирующих
//                  пользователя
////////////////////////////////////////////////////
                        {
        int             iCode;
        char            cBuf[LN_PAROL+1];
        switch (message)
 {
                case WM_COMMAND:
{
//-------------------------------------------------
// Обработка нажатия кнопки подтверждения выбора
//                      данных
//-------------------------------------------------
   if (wParam == ID_BT_1)
                        {
// Анализ введенной пользователем информации
   if ( SendMessage
   (hWndEd_1, WM_GETTEXTLENGTH, 0, 0L)) {   // *
 SendMessage(hWndEd_1, WM_GETTEXT, LN_PAROL+1,
 (LPSTR)cBuf); AnsiUpper(cBuf);
 SetWindowText(hWndEd_1, NULL);
                if( lstrcmp("PLOT", cBuf) == 0 )
    { DestroyWindow(hWnd); byFlag=1;
 ShowWindow(hWndSt, SW_SHOWMAXIMIZED); return NULL;
    }
 OemToAnsi(
 "Несоответствие идентифицирующих данных!"
 "Продолжите работу с программой или прекратите"
 " ее выполнение.", cText);
  iCode=MessageBox(hWnd, cText, NULL, MB_RETRYCANCEL);
           if( iCode == IDRETRY) SetFocus(hWndEd_1);
                                  else {
  DestroyWindow(hWnd); return NULL;   }
                                        }     // *
                                        else {
 OemToAnsi(
 "Не были введены идентифицирующие данные", cText);
 MessageBox(hWnd, cText, NULL, MB_OK);
 SetFocus(hWndEd_1); return NULL;             }
                        } else break;
}
 }
return(DefWindowProc(hWnd, message, wParam, lParam));
                        }
    void CrWndParole(HWND hWnd)
////////////////////////////////////////////
// Создание и отображение окна ввода пароля
////////////////////////////////////////////
{
 WndClass.lpszClassName="DOSTUP";
 WndClass.hInstance=hInstance_;
 WndClass.lpfnWndProc=(FARPROC) WndProcShell_p_1;
 WndClass.style=NULL;
 WndClass.hbrBackground=GetStockObject(LTGRAY_BRUSH);
 WndClass.hCursor=LoadCursor(NULL, IDC_ARROW);
 WndClass.hIcon=NULL;
 WndClass.lpszMenuName=NULL;
 WndClass.cbClsExtra=NULL;
 WndClass.cbWndExtra=NULL;
 RegisterClass(&WndClass);
 OemToAnsi("Ввод идентифицирующих данных", cText);
 hWnd_1=CreateWindow("DOSTUP", cText,
 WS_POPUP : WS_THICKFRAME : WS_CAPTION,
 tm.tmMaxCharWidth*8, iHs*6, tm.tmMaxCharWidth*22,
 iHs*7, hWnd, NULL, hInstance_, NULL);
 hWndEd_1=CreateWindow("Edit", (LPSTR)NULL,
 WS_CHILD : ES_LEFT : WS_BORDER : WS_VISIBLE :
 ES_PASSWORD, tm.tmMaxCharWidth*4, iHs*1,
 tm.tmMaxCharWidth*LN_PAROL+1, iHs+4,
 hWnd_1, ID_ED_1, hInstance_, NULL);
//  Создание окна подтверждения ввода данных
 hWndBt_1=CreateWindow("button", "OK",  WS_CHILD :
 BS_DEFPUSHBUTTON : WS_VISIBLE, tm.tmMaxCharWidth*8,
 iHs*3, tm.tmMaxCharWidth*4, iHs+6, hWnd_1, ID_BT_1,
 hInstance_, NULL); ShowWindow(hWnd_1, SW_SHOWNORMAL);
 UpdateWindow(hWnd_1); SetFocus(hWndEd_1);
}
```

■

Листинг 2. Текст задачи «Приложение 1»

```
#include        <windows.h>
#define         ID_ED   11
#define         ID_BT   12
#define         LN_TEXT 12
long FAR PASCAL WndProcAppl_1
                (HWND, unsigned, WORD, LONG);
LPSTR   lpAdrMem;
```

```
HWND    hWndEd, hWndBt;
int     iHs;
char    cText[100];
HDC     hDC;
TEXTMETRIC      tm;
int     PASCAL  WinMain(HANDLE hInstance,
HANDLE  hPrevinstance, LPSTR lpCmdLine, int nCmdShow)
{
HWND    hWnd;
HWND    hMainWnd;       // Индекс родительского окна
MSG     message;
WNDCLASS        WndClass;
// Дешифрация блока параметров, получаемых
//       от родительской программы
asm     mov     bx,word ptr lpCmdLine+2;
asm     mov     es,bx;
asm     mov     di,word ptr lpCmdLine;
asm     mov     ax,di;
asm     add     ax,2;
asm     cmp     byte ptr es:[di],1;
asm     jz      ap_1;
asm     mov     bl,byte ptr es:[di+1];
asm     add     di,2;
ap_2:
asm     cmp     byte ptr es:[di],0;
asm     jz      ap_1;
asm     cmp     byte ptr es:[di],bl;
asm     jnz     ap_3;
asm     mov     byte ptr es:[di],0
ap_3:
asm     inc     di;
asm     jmp     ap_2;
ap_1:
asm     mov     di,ax;
asm     mov     ax,word ptr es:[di];
asm     mov     word ptr hMainWnd,ax;
asm     mov     ax,word ptr es:[di+2];
asm     mov     word ptr lpAdrMem,ax;
asm     mov     ax,word ptr es:[di+4];
asm     mov     word ptr lpAdrMem+2,ax;
        if (!hPrevinstance)
                                        {
// Создание основного окна
 WndClass.lpszClassName="APPL_1";
 WndClass.hInstance=hInstance;
 WndClass.lpfnWndProc=(FARPROC) WndProcAppl_1;
 WndClass.style=NULL;
 WndClass.hbrBackground=GetStockObject(GRAY_BRUSH);
 WndClass.hCursor=NULL;
 WndClass.hIcon=LoadIcon(NULL, IDI_APPLICATION);
 WndClass.lpszMenuName=NULL; WndClass.cbClsExtra=NULL;
 WndClass.cbWndExtra=NULL; RegisterClass(&WndClass);
 OemToAnsi("Приложение 1", cText); hWnd=CreateWindow(
 "APPL_1", cText, WS_CHILD | WS_CAPTION | WS_SYSMENU |
 WS_MINIMIZEBOX | WS_CLIPSIBLINGS, CW_USEDEFAULT,
 CW_USEDEFAULT, 300, 200, hMainWnd, NULL,
 hInstance, NULL);
 hDC=GetDC(hWnd); GetTextMetrics(hDC, &tm);
 ReleaseDC(hWnd, hDC);
 iHs=tm.tmExternalLeading+tm.tmInternalLeading+
 tm.tmHeight;
// Создание окна ввода информации
 hWndEd=CreateWindow("Edit", (LPSTR)NULL,
 WS_CHILD | ES_LEFT | WS_BORDER | WS_VISIBLE,
 tm.tmMaxCharWidth*4, iHs*3,
 tm.tmMaxCharWidth*LN_TEXT+1, iHs+6,
 hWnd, ID_ED, hInstance, NULL);
//  Создание окна подтверждения ввода данных
 hWndBt=CreateWindow("button", "OK",
 WS_CHILD | BS_DEFPUSHBUTTON | WS_VISIBLE,
 tm.tmMaxCharWidth*8, iHs*6, tm.tmMaxCharWidth*4,
 iHs+6, hWnd, ID_BT, hInstance, NULL);
 (BYTE)(*lpAdrMem)=0; // Установили признак отсутствия
                      // информации в общей
                      // области
 ShowWindow(hWnd, SW_SHOWMINIMIZED);
                                      UpdateWindow(hWnd);
while (GetMessage(&message,NULL,0,0))
        {
TranslateMessage(&message); DispatchMessage(&message);
        }
return  message.wParam;
                                              }
        return  NULL;
}
long    FAR     PASCAL WndProcAppl_1(
HWND hWnd, unsigned message, WORD wParam, long lParam)
                        {
        int     iMode;
        PAINTSTRUCT     ps;
        switch (message)
{
                case WM_PAINT:
 {
    hDC=BeginPaint(hWnd, &ps);
 iMode=GetBkMode(hDC); SetBkMode(hDC, TRANSPARENT);
 OemToAnsi("Введите символьную информацию", cText);
 TextOut(hDC, tm.tmMaxCharWidth*2, iHs, cText,
 lstrlen(cText)); SetBkMode(hDC, iMode);
 EndPaint(hWnd, &ps); SetFocus(hWndEd);
 return NULL;
  }
                case WM_COMMAND:
//-------------------------------------------------
// Обработка нажатия кнопки подтверждения выбора
//                      данных
//-------------------------------------------------
   if (wParam == ID_BT)
                        {
// Анализ введенной пользователем информации
   if ( SendMessage
   (hWndEd, WM_GETTEXTLENGTH, 0, 0L)) {
  SendMessage(hWndEd, WM_GETTEXT, LN_TEXT*2, lpAdrMem);
       SetWindowText(hWndEd, NULL); return NULL;
                                        }
                        }
}
return(DefWindowProc(hWnd, message, wParam, lParam));
     }
```

Листинг 3. Текст задачи «Приложение 2»

```
#include        <windows.h>
long FAR PASCAL WndProcAppl_2
                (HWND, unsigned, WORD, LONG);
LPSTR   lpAdrMem;
int     iHs;
char    cText[100];
HDC     hDC;
TEXTMETRIC      tm;
int     PASCAL  WinMain(HANDLE hInstance,
HANDLE  hPrevinstance, LPSTR lpCmdLine, int nCmdShow)
{
HWND    hWnd;
HWND    hMainWnd;       // Индекс родительского окна
MSG     message;
WNDCLASS        WndClass;
// Дешифрация блока параметров, получаемых
//       от родительской программы
```

```
   asm     mov     bx,word ptr lpCmdLine+2;
   asm     mov     es,bx;
 asm     mov     di,word ptr lpCmdLine;
 asm     mov     ax,di;
 asm     add     ax,2;
 asm     cmp     byte ptr es:[di],1;
 asm     jz      ap_1;
 asm     mov     bl,byte ptr es:[di+1];
 asm     add     di,2;
 ap_2:
 asm     cmp     byte ptr es:[di],0;
 asm     jz      ap_1;
 asm     cmp     byte ptr es:[di],bl;
 asm     jnz     ap_3;
 asm     mov     byte ptr es:[di],0
 ap_3:
 asm     inc     di;
 asm     jmp     ap_2;
 ap_1:
 asm     mov     di,ax;
 asm     mov     ax,word ptr es:[di];
 asm     mov     word ptr hMainWnd,ax;
 asm     mov     ax,word ptr es:[di+2];
 asm     mov     word ptr lpAdrMem,ax;
 asm     mov     ax,word ptr es:[di+4];
 asm     mov     word ptr lpAdrMem+2,ax;
        if (!hPrevinstance)
                                        {
 //              Создание основного окна
  WndClass.lpszClassName="APPL_2";
  WndClass.hInstance=hInstance;
  WndClass.lpfnWndProc=(FARPROC) WndProcAppl_2;
  WndClass.style=NULL;
  WndClass.hbrBackground=GetStockObject(LTGRAY_BRUSH);
  WndClass.hCursor=NULL;
  WndClass.hIcon=LoadIcon(NULL, IDI_APPLICATION);
  WndClass.lpszMenuName=NULL; WndClass.cbClsExtra=NULL;
  WndClass.cbWndExtra=NULL; RegisterClass(&WndClass);
  OemToAnsi("Приложение 2", cText); hWnd=CreateWindow(
  "APPL_2", cText, WS_CHILD | WS_CAPTION | WS_SYSMENU |
  WS_MINIMIZEBOX | WS_CLIPSIBLINGS, CW_USEDEFAULT,
  CW_USEDEFAULT, 340, 120, hMainWnd, NULL,
  hInstance, NULL);
  hDC=GetDC(hWnd); GetTextMetrics(hDC, &tm);
  ReleaseDC(hWnd, hDC);
  iHs=tm.tmExternalLeading+tm.tmInternalLeading+
  tm.tmHeight;
  ShowWindow(hWnd, SW_SHOWMINIMIZED);
                                        UpdateWindow(hWnd);
while (GetMessage(&message,NULL,0,0))
        {
TranslateMessage(&message); DispatchMessage(&message);
        }
return  message.wParam;
                                        }
        return  NULL;
}
long    FAR     PASCAL WndProcAppl_2(
HWND hWnd, unsigned message, WORD wParam, long lParam)
                        {
        int     iMode;
        PAINTSTRUCT     ps;
        switch (message)
{
                case WM_QUERYOPEN:
{
if ( (BYTE)(*lpAdrMem) == 0 ) {
 OemToAnsi("Отсутствуют данные в общей памяти", cText);
 MessageBox(hWnd, cText, NULL, MB_OK); return NULL;
                                } else break;
}
                case WM_SHOWWINDOW:
{
if ( (BYTE)(*lpAdrMem) == 0 ) {
   if(wParam != 0){
                        ShowWindow(hWnd, SW_SHOWMINIMIZED);
 OemToAnsi("Отсутствуют данные в общей памяти", cText);
 MessageBox(hWnd, cText, NULL, MB_OK); return NULL;
                        }
                                } else break;
}
                case WM_PAINT:
 {
         hDC=BeginPaint(hWnd, &ps);
 iMode=GetBkMode(hDC); SetBkMode(hDC, TRANSPARENT);
 OemToAnsi("Из приложения 1 была получена строка:",
 cText);
 TextOut(hDC, tm.tmMaxCharWidth*2, iHs, cText,
 lstrlen(cText));
 TextOut(hDC, tm.tmMaxCharWidth*2, iHs*3, lpAdrMem,
 lstrlen(lpAdrMem)); SetBkMode(hDC, iMode);
 EndPaint(hWnd, &ps); return NULL;
 }
                case WM_MOVE:
                case WM_LBUTTONDOWN:
                case WM_WINDOWPOSCHANGED:
                case WM_WINDOWPOSCHANGING:
 {
   InvalidateRect(hWnd, NULL, TRUE);
   UpdateWindow(hWnd); return NULL;
 }
}
return(DefWindowProc(hWnd, message, wParam, lParam));
                        }
```

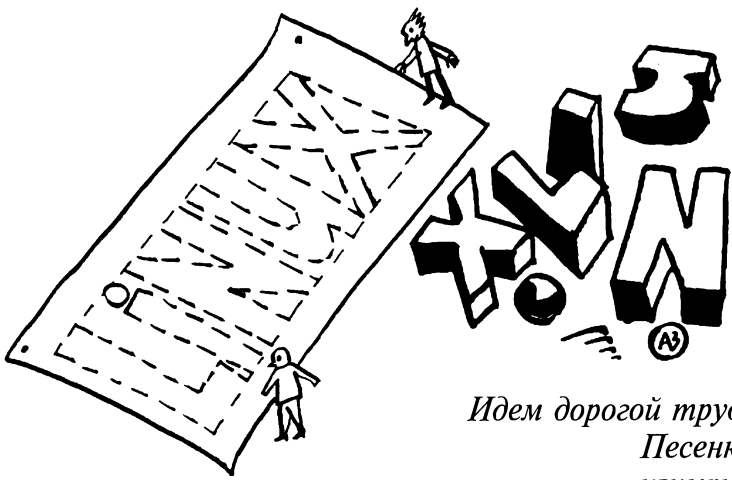
■

Листинг 4. Тексты файлов определения модулей

```
NAME            SHELL
DESCRIPTION     'Оболочка для запуска задач'
EXETYPE         WINDOWS
CODE            MOVEABLE
DATA            MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE       8192
EXPORTS
                WndProcShell_p          @1
                WndProcShell_p_1        @2
NAME            APPL_1
DESCRIPTION     'Приложение 1'
EXETYPE         WINDOWS
CODE            MOVEABLE
DATA            MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE       8192
EXPORTS
                WndProcAppl_1 @1
NAME            APPL_2
DESCRIPTION     'Приложение 2'
EXETYPE         WINDOWS
CODE            MOVEABLE
DATA            MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE       8192
EXPORTS
                WndProcAppl_2 @1
```

■

Как без головной боли и нервотрепки установить Linux



В. Водолазкий

Идем дорогой трудной, дорогой непрямой...
Песенка из мультфильма «Волшебник изумрудного города»

Часть 2

Конечно, начинать новый цикл статей надо с нового года. Поэтому я приношу свои извинения читателям за то, что им придется (я надеюсь!) добывать последний номер «Монитора» за прошлый год, в котором мы начали наш рассказ об установке операционной системы Linux [1]. В этой статье мы продолжим рассказ об установке системы, а именно — рассмотрим, как на самом деле устанавливается графическая оболочка XWindows.

Но прежде всего давайте рассмотрим, что же представляет собой XWindows? По большому счету, это набор протоколов, регламентирующих взаимодействие системы «клиент-сервер» (то есть программ «X-сервер» и «X-клиент»). *X-клиент* представляет собой аппаратно-независимую прикладную программу, которая использует обращения к серверу. X-сервер управляет выводом поступающих от клиента данных на экран и вводом данных от пользователя, которые передаются текущему активному приложению-клиенту. Находящиеся в памяти клиенты друг другу не мешают, хотя и могут обмениваться сообщениями. Конечно, искушенный читатель сразу же начнет искать аналогии с MS Windows, однако уместно заметить, что XWindows, во-первых, заметно старше, а во-вторых, эксплуатируется по меньшей мере на нескольких десятках вычислительных платформ. Естественно, что приложения-клиенты по большей части переносимы на уровне исходного кода.

Linux поставляется со всеми библиотеками и инструментальными средствами, необходимыми для разработки приложений для XWindows. Так что у вас есть шанс отличиться. Хотя одна задача уже решена! Г-н Вакуленко разработал пакет для русификации XWindows, который вы можете найти на третьем диске «Trans-Ameritech», поддерживающий различные кодовые таблицы и раскладки клавиатуры. Возможно, кому-то покажется относительно бедным набор шрифтов, но, как вы догадываетесь, здесь есть к чему приложить руки.

Но для того чтобы заняться программированием в XWindows, оболочку надо, по крайней мере, установить. Вот этим мы сейчас и займемся.

Прежде всего, необходимо установить на винчестер все пакеты, связанные с XWindows. Конечно же, к ним относятся X и XAP, но не забудьте и библиотеки и вспомогательные средства, например OI. Но с этой проблемой, я надеюсь, вы справитесь. Самое важное — не скупитесь, ставьте по максимуму! Да не забывайте о тонкостях, которые нигде не описаны.

Тонкость номер 1 — X-сервер

При установке ядра системы вам будет предложено установить один из семи уже скомпилированных серверов.

* 8514a предназначен для платы адаптера IBM 8514a и носит, прямо скажем, абстрактный характер. Во-первых, эти видеоплаты в России встречаются исключительно редко. А во-вторых, адаптер реализован для шины MCA (мне, по крайней мере, другие варианты не попадались), которая не поддерживается самой системой Linux.

* Monochrome X-server ориентирован на использование VGA-адаптера с монохромным монитором. Но имейте в виду, никаких градаций яркости! Этот сервер признает всего два цвета — черный и белый, зато отображает данные максимально быстро.
* SVGA server — вот он, пожалуй, будет использоваться чаще всего, ведь SVGA-платы получают все большее распространение. Но необходимо отметить, что меньше всего проблем у вас возникнет с платой Trident 8900 (буквы роли не играют). А вот плата Trident 9000 имеет ряд дефектов, которые порой приводят к «рассыпанию» изображения на экране или «зависанию» компьютера. Поэтому будьте бдительны! SVGA-server поддерживает режим 256 цветов (разрешение определяется возможностями монитора и объемом видеопамяти на плате).
* VGA server поддерживает только 16 цветов. Предназначен для использования с устаревшими платами VGA-адаптеров, имеющими 256 кб видеопамяти. Поскольку прогресс в области аппаратных средств движется семимильными шагами, этот сервер так и остался на уровне эксперимента. Используйте его только в самом крайнем случае. Результаты, вообще говоря, непредсказуемы.
* Оставшиеся три сервера ориентированы на использование наиболее популярных плат с ускорителями класса Windows Accelerator. Поскольку интерфейс пользователя в XWindows также построен на основе прямоугольников, как и MS Windows, использование этих плат позволяет достичь значительного выигрыша в производительности.

Выбирайте для установки только один сервер, который соответствует вашему видеоадаптеру. Тратить место на диске на остальные смысла не имеет. Кроме того, процедура установки автоматически перенастроит указатели (символьные связи) на последний установленный сервер (в списке как раз не до конца отлаженный VGA-сервер).

Тонкость номер 2 — конфигурирование системы

После того как вы запустите XWindows, вы с удивлением обнаружите, что скорость вывода графики в полтора-два раза выше, чем в MS Windows. Но чтобы добиться столь впечатляющих результатов, вам придется немного попотеть. Дело в том, что сервер сам по себе предназначен для обслуживания *семейства* видеоадаптеров, а для практической работы вы должны настроить его на работу с конкретной парой видеоадаптер-монитор.

Эти настройки размещаются в файле Xconfig, который должен находиться в каталоге /usr/X11/lib/X11. Типовой файл, ориентированный на видеоплату Trident 8900, содержится в этом же каталоге в файле Xconfig.Trident8900c. Достаточно просто выполнить команду:

```
mv Xconfig.Trident8900c Xconfig
```

И вся ваша настройка (если конечно, у вас плата Trident 8900) на этом закончена. В этом случае можно считать, что вам крупно повезло. Но рассмотрим менее приятный случай — у вас другой тип видеоадаптера. Задача состоит в том, чтобы определить видеорежимы, которые будет поддерживать XWindows и, что самое важное, задать *корректные* режимы, которые поддерживаются как видеоадаптером, так и монитором.

Но вначале давайте рассмотрим, что же представляет из себя файл Xconfig. Вот пример типичного файла:

```
RGBPath     "/usr/X386/lib/X11/rgb"
FontPath    "/usr/lib/X11/fonts/75dpi/,/usr/lib/X11/fonts/Speedo/"
Keyboard
   AutoRepeat 500 5
   ServerNumLock
LeftAlt  Meta
RightAlt Compose
ScrollLock ModeLock
Microsoft "/dev/mouse"
   BaudRate 1200
vga256
   Virtual 1024 1024
   ViewPort      0 0
   Modes         "800x600" "640x480" "1024x768i"
   Chipset "tvga8900c"
Displaysize     800 600
Clocks 25.30 28.32 45.00 36.00 57.30 65.10 50.40 39.90
ModeDB
# name      clock  horizontal timing     vertical timing      flags
"640x480"   25     640  672  768  800    480  490  492  525
"800x600"   36     800  840  912  1024   600  600  602  625
"1024x768i" 45     1024 1048 1224 1264   768  768  776  816   Interlace
```

Давайте начнем по порядку. Первые строки, вплоть до описания конфигурации «мыши», вам, скорее всего, изменять не придется. Как пути, так и настройки клавиатуры выглядят вполне приемлемо. Более того, они даже не вступают в конфликт с русификатором! Что касается «мыши», то настройка манипулятора, вообще говоря, осуществляется программой setup, а в Xconfig вы, как правило, только указываете ее тип — это касается нашего случая, а также наиболее распространенного — «Microsoft».

Кстати, если вы используете «мышь» типа PS/2, как, например, на машинах фирмы «Acer», не забудьте включить ее поддержку при компиляции ядра Linux. Иначе толку никакого не будет. Более подробную информацию о конфигурировании «мыши» вы можете получить из встроенной системы помощи:

```
man Xconfig
```

Но с этим вдумчивый читатель справится и без меня. Я постараюсь по возможности понятно рассказать, что делать дальше. В файле конфигурации мы видим секцию vga256. Пора взглянуть на нее внимательнее.

```
Virtual 1024 1024
```

Это виртуальный размер экрана. Если вы работали с пакетами Word for Windows или хотя бы рисовали картинки каким-либо графическим редактором, то понятие виртуального экрана вам знакомо. Числа обозначают разрешение, которое определяется объемом видеопамяти. Для видеоплаты Trident 8900 одного мегабайта достаточно как раз для хранения образа экрана 1024*1024. Платы с ускорителями расходуют память куда более агрессивно, а поэтому, если вы рассчитываете на действительно выдающиеся результаты, подумайте над расширением видеопамяти плат класса S3 хотя бы до 2 Мб.

```
ViewPort 0 0
```

Это начальное смещение рабочего окна относительно левого верхнего угла виртуального экрана. Оставляйте как есть.

```
Modes
```

В данной строке вы просто перечисляете видеорежимы, которые будет использовать ваш X-сервер. Конечно, инициализируется только один из них, но вот какой? Да самый первый в списке. А следующая строка:

```
DisplaySize 800 600
```

означает, что X-сервер инициализирует (вне зависимости от текущего разрешения) адресуемую область экрана 800 на 600.

```
Chipset
```

Здесь указывается название набора микросхем, на базе которого изготовлен ваш видеоадаптер. Список поддерживаемых наборов вы найдете с помощью системы man, а узнать, какой адаптер у вас, можете, прочитав документацию к видеоплате.

Ну, а следующая строка выглядит жутковато. Цифры, на первый взгляд, совершенно непонятные. Означают они настройки временных параметров видеоадаптера и изменяются от платы к плате. Но, получить эти «тайные знаки» можно относительно просто. Вам нужно «обрезать» Xconfig по строку Clocks включительно и запустить следующую команду:

```
X -probeonly
```

Система просмотрит все диапазоны возможных значений настройки таймеров видеоадаптеров и выдаст на экран список чисел, которые вы должны в том же порядке перенести в файл Xconfig. Не беда, если они займут несколько строк. А кроме того, обратите внимание на максимальную пропускную полосу монитора. На хороших моделях она достигает 90 МГц. Это абстрактное значение очень важно для нас. Но позвольте вначале небольшое лирическое отступление.

Хочу сразу же предупредить, что изложенное далее представляет собой мои досужие домыслы и не может восприниматься как антиреклама «пострадавших» фирм.

Вы смотрите телевизор? Не задавались ли вы вопросом, почему иногда изображение на дисплее, которое вы видите на телевизионном экране, «стоит» как влитое, а иногда по нему бегут какие-то полосы? И почему изображение на экране лишено искажений столь редко?

Да, я слышал о популярных вариантах ответа. Но позвольте предложить вам свое решение. Во-первых, вспомните школьный курс физики, а именно — фигуры Лиссажу. То, что мы видим на экране, — тоже разновидность биений, возникающая в тех случаях, когда кадровая развертка монитора компьютера близка или сопоставима с кадровой разверткой видеокамеры. Так вот, пусть глаз не замечает смены кадров на мониторе, видеокамера безжалостно демонстрирует нам *низкую* частоту развертки вашего компьютера. Кстати, почему экраны рабочих станций SUN или DEC получаются на ТВ вполне прилично? Да потому, что в этих мониторах частота смены кадров выше 70 Гц, в то время как видеокамера делает около 30 кадров в секунду!

Между тем, чем выше частота развертки, тем стабильнее «стоит» на экране изображение и меньше устают глаза. Но при этом на монитор поступает значительно больший поток данных, а следовательно, увеличивается и эквивалентная пропускная полоса.

Что из этого следует? А вот что! Увеличение частоты кадровой развертки при сохранении заданного разрешения способствует:

а) максимально полному использованию возможностей ваших аппаратных средств;
б) повышению стабильности изображения на экране.

К счастью, XWindows предоставляет вам уникальные возможности в оптимизации настройки аппаратных средств. А для этого потребуется изменить параметры видеорежимов.

Но сказать легче, чем сделать. Вначале необходимо попробовать пойти путем минимального сопротивления. Обратите ваше внимание на каталог /usr/X11/lib/X11/etc. И прежде всего — на файл modeDB.txt. А посмотреть его содержимое можно командой

```
cat modeDB.txt | less
```

(Для начинающих: cat копирует файл в выходной поток, а less подхватывает входной поток, буферизирует его и позволяет просмотреть его содержимое, подобно хорошо знакомой вам программе README).

Этот файл представляет собой базу данных, составленную любителями Linux со всего мира, и состоит из записей следующего вида:

```
# Monitor:              Seiko CM 1440
# Contributor:          Jeff Jennings [jennings@stortek.com]
# Last Edit Date:       12/9/91
#
# name          clock  horizontal timing     vertical timing      flags
"1024x768i"     45     1024 1064 1224 1264    768  777  785  817  Interlace
"1152x900i"     45     1152 1184 1288 1360    900  898  929  939  Interlace
```

Ваша задача состоит в том, чтобы заменить содержимое строк описания режимов в файле Xconfig на найденные вами в таблице значения. Но позвольте вам еще раз напомнить: тип монитора играет очень важную роль!

Вообще говоря, не стоит проявлять излишней самодеятельности. Взгляните на таблицу 1, в которой перечислены общие режимы, которые поддерживают почти все мониторы. Но не поленитесь, откройте книжечку, поставляемую вместе с вашим компьютером. Возможны два варианта. Если вы не экономили, то у вас окажется монитор класса Multisync, поддерживающий *любую* частоту кадровой и строчной развертки в заданном диапазоне. А если ваш компьютер приобретен на «распродаже», будут доступны только несколько режимов, из которых и придется выбирать. Впрочем, мы не будем превращать «Монитор» в справочное пособие. Список настроек видеорежимов (частоты развертки и настройки таймеров для горизонтальной и вертикальной разверток) вы сможете найти в файле /usr/X11/lib/X11/etc/README.config. А если вы все же решитесь заняться оптимизацией вручную, то имеет смысл обратиться к файлу Videomodes.doc в том же каталоге. Кроме того, полезную информацию вы сможете найти и в каталоге

Таблица 1

Режим	Частота кадров	Синхронизация строк	Эквивалентная полоса	Черезстрочная развертка?	Поддержка VESA?
640×480	60 Hz	31.5 kHz	25.175 MHz	Нет	Нет
640×480	60 Hz	31.5 kHz	25.175 MHz	Нет	Нет
640×480	63 Hz	32.8 kHz	28.322 MHz	Нет	Нет
640×480	70 Hz	36.5 kHz	31.5 MHz	Нет	Нет
640×480	72 Hz	37.9 kHz	31.5 MHz	Нет	Да
800×600	56 Hz	35.1 kHz	36.0 MHz	Нет	Да
800×600	56 Hz	35.4 kHz	36.0 MHz	Нет	Нет
800×600	60 Hz	37.9 kHz	40.0 MHz	Нет	Да
800×600	60 Hz	37.9 kHz	40.0 MHz	Нет	Нет
800×600	72 Hz	48.0 kHz	50.0 MHz	Нет	Да
1024×768i	43.5 Hz	35.5 kHz	44.9 MHz	Да	Нет
1024×768	60 Hz	48.4 kHz	65.0 MHz	Нет	Да
1024×768	60 Hz	48.4 kHz	62.0 MHz	Нет	Нет
1024×768	70 Hz	56.5 kHz	75.0 MHz	Нет	Да
1024×768	70 Hz	56.25 kHz	72.0 MHz	Нет	Нет
1024×768	76 Hz	62.5 kHz	85.0 MHz	Нет	Нет
1280×1024i	44 Hz	51 kHz	80.0 MHz	Да	Нет
1280×1024i	44 Hz	47.6 kHz	75.0 MHz	Да	Нет
1280×1024	59 Hz	63.6 kHz	110.0 MHz	Нет	Нет
1280×1024	61 Hz	64.24 kHz	110.0 MHz	Нет	Нет
1280×1024	74 Hz	78.85 kHz	135.0 MHz	Нет	Нет

/usr/doc/faq/howto. Для просмотра файлов без их распаковки рекомендую воспользоваться командой:

```
zcat <имя файла > : less
```

(Команда zcat осуществляет копирование сжатого входного файла на устройство стандартного вывода с автоматический распаковкой.)

Тонкость номер 3 — запуск XWindows

Но я надеюсь, вы справитесь с настройкой. Теперь самое время попробовать, что получилось. Запускать XWindows необходимо с помощью openwin или startx. Перед вами появится почти чистый экран. Нажмите кнопку «мыши». А теперь вторую. Перед вами появится меню, которое, кстати, вы можете отредактировать по своему усмотрению, как и системное меню, которое по умолчанию содержит команды Close, Restore, Kill и так далее.

А теперь вы можете заняться исследованием XWindows так же, как в свое время приступали к MS Windows. Поверьте, многое даже проще!

Тонкость номер 4, но не последняя — русификация

Как я уже указывал, проблема русификации XWindows решена г-ном Вакуленко. И с моей стороны было бы просто плагиатом перепечатывать его инструкцию к пакету хсуг. Тем более, что она написана на русском языке.

Но одна сложность все же есть. А именно — русификация Linux в текстовом режиме. Мне помогли в этом специалисты из фирмы СКТБ Компьютерных Сетей, которая использует Linux при создании автоматизированных систем обработки информации. Кстати, именно поэтому все компьютеры, выпускаемые СКТБ, поддерживают Linux, как говорится, в полном объеме. Так вот, чтобы заставить Linux говорить по-русски, вам потребуется загрузить ЙЦУКЕН-раскладку и знакогенераторы кириллицы с помощью программы Setup (Remap your keyboard), а затем выполнить такую последовательность команд, которую можно оформить как script-файл:

```
loadkeys /usr/lib/kbd/keytables/russian.map
mapscrn koi2alt
setfont Cyr_a8x16
printf "\033(K" echo  Для включения кириллицы нажмите правый Alt
```

А после этого вам остается лишь следовать инструкциям.

И в заключение. Я отлично понимаю, что знаю далеко не все тонкости об XWindows. А поэтому обращаюсь к читателям нашего журнала с просьбой сообщать обо всех моих ошибках и неточностях, а также о ваших находках. XWindows стоит того, чтобы заняться ей всерьез! ■

Литература

1. **В. Водолазкий. Как без головной боли и нервотрепки установить Linux// Монитор. — 1994. — № 8.**

Новое поколение выбирает UNIX!

Н.Кожекин

Недавно я прочитал в журнале «Монитор» статью В. Водолазкого [1]. В ней утверждалось, что «молодое поколение, видимо, всерьез полагает, что, кроме MS-DOS, никаких других операционных систем не бывает...». Признаться, такое заявление меня обескуражило. Ведь я, собственно говоря, и есть представитель этого молодого поколения. Тогда мне захотелось что-то ответить, и эта статья — попытка ответа.

Дороги, которые мы выбираем

Одним из самых популярных производителей компьютеров для работы UNIX является фирма HP, как видно из таблицы 1. Поэтому мы начнем именно с ее реализации UNIX — Hewlett-Packard OpenView. Немаловажным фактором является то, что фирма «SUN Microsystems» лидирует на финансовом и коммерческом рынке, в то время как фирма «Hewlett-Packard» в основном известна в научных и инженерных кругах, хотя отмечены и попытки HP пробраться в сферу коммерческих разработок.

Обычно говорят, что UNIX-компьютеры стоят дорого. Однако довольно часто использование решений UNIX может привести к значительной экономии средств. Ведь при работе в сети здесь используются процессор и память сервера, а не только винчестер, как при работе в Novell.

Второе важное преимущество UNIX связано с тем, что это изначально многозадачная многопользовательская операционная система. В UNIX с каждым пользователем связаны личный и групповой идентификаторы. Их система передает порождаемым пользователем процессам, а те, в свою очередь, - наследникам (созданный любым процессом файл получает эффективные идентификаторы процесса). Вообще каждый файл на диске обладает упомянутой парой идентификаторов. Права процессов для доступа к файлу хранятся в атрибутах защиты файла. Файл может быть считан, записан, выполнен. Таким образом, процессы могут принадлежать владельцу исполняемого файла, членам его группы и остальным пользователям. Процесс может также иметь зависящие от реализации привилегии. Чаще всего планировщик UNIX (он же супервизор — **прим. ред.**) использует политику разделения времени, что для большинства задач хорошо. Однако некоторые разновидности UNIX поддерживают и реальный режим, как, например, описанная в статье [3] система QNX или UNIX фирмы USL. Таким образом, потратив большие деньги на сервер, вы можете вполне ощутимо сэкономить на дешевых терминалах.

(Позвольте вставить несколько строк от редактора. Предыдущий абзац может показаться достаточно трудным для понимания, если вы до сих пор работали только с MS-DOS. Однако мы сочли возможным обойтись без дальнейших разъяснений в этой статье. В конце концов, о системном администрировании в UNIX и о работе супервизора мы расскажем отдельно. — **Прим. ред.**)

Существует также круг задач, при которых использование мощностей, предоставляемых UNIX, просто необходимо. Это, например, физические исследования, требующие быстрой обработки большого количества данных. Однако поскольку большинство пользователей компьютеров — это различные клерки, UNIX им реально не нужна и непонятна, и они обходятся MS-DOS.

Система UNIX ориентирована в первую очередь на профессионалов. Хотя и «любителям» может пригодиться высокая степень защиты и надежности данных, присущая UNIX. Кроме того, не каждому по карману иметь дома компьютер под управлением UNIX. Но вот тут конкуренцию MS-DOS могут составить уже не коммерческие версии UNIX, а Linux, как и было доказано в [1].

Преимущества и недостатки

UNIX является открытой многозадачной многопроцессорной операционной системой вытесняющей многозадачности. Его графическая оболочка — XWindows представляет собой мощную графическую систему работы X-сервера с X-клентом через X-протокол. Причем, в отличие от MS Windows, она полностью уважает независимость и равноправие всех X-клиентов: ни один X-клиент не вмешивается в работу другого, за исключением той ситуации, когда они взаимодействуют непосредственно. За время своего существования UNIX-мир стал довольно разнообразным. Поэтому для него разработано множество стандартов, которые подробно описаны в статье [2].

Как к достоинствам, так и к недостаткам UNIX можно отнести тот факт, что она является одновременно дисковой и

Таблица 1

Продажи 32-битовых операционных систем в мире в 1993 году

Компания/продукт	Продажи
«Sun Microsystems»/SunNet Manager	31 %
«Hewlett-Packard»/OpenView	25,2 %
«NetLabs»/DiMons	9,5 %
«Cableton Systems»/Spectrum	8,6 %
IBM/NetView/6000	5 %
Другие	20,7 %

Источник: «International Data Corp.»

сетевой операционной системой. Многие UNIX-системы обладают также следующим недостатком: при аварийном выключении питания сервера часть информации безвозвратно теряется, что иногда приводит к повреждению всего дискового тома. Но некоторые современные версии лишены подобных проблем, например UNIXWare фирмы «Novell». Зато достоинством является возможность получения посмертного дампа программы, в большинстве случаев оказывающего существенную помощь при отладке программ и диагностировании причин сбоев. А еще следует отметить, что «зависания» в UNIX происходят гораздо реже, чем в DOS, что обеспечивается именно за счет многозадачности: переключившись на другую виртуальную линию, вы можете без проблем выгрузить «зависший» процесс из памяти. И накладок, характерных для MS Windows, при этом не возникает.

Таблица 2

Цены на 32-битовые UNIX-системы для ПК на базе процессоров Intel

Компания	Продукт	Цена по каталогу в $ USA			Минимальные требования к аппаратуре
		Базовый пакет	Пакет без ограничения числа пользователей	Пакет разработчика	
«SunSoft, Inc.»	Interactive Unix 4.0	495	895	795(SDK)	386SX RAM-8, HDD-120
«NeXT, Inc.»	Nextstep 3.2	795	-	2995	486 RAM-16, HDD-200
«Novell, Inc.»	Novell UNIXWare 1.1	249	1299	99(SDK)	386DX RAM-12, HDD-100
«The Santa Cruz Operation, Inc.»	SCO Open Desktop 3.0	1195	3195	1595	386 RAM-8, HDD-110
«SunSoft, Inc.»	Solaris for x86 2.1	795	5995	995(SDK)	386DX RAM-16, HDD-200*

* — требуется математический сопроцессор.

SDK — Software Development Kit

Таблица 3

Где используют Linux?

Используют сами	Знают о других пользователях	Итого	% от общего количества	Место
6476	102	6578	75%	Дом
2073	61	2134	24%	Работа
291	521	812	9%	Школа
679	55	734	8%	Еще где-то
62	0	62	0%	Не используют
ВСЕГО 7953	716	8669	100%	

Замечание. Некоторые люди используют Linux во многих местах, так что ВСЕГО не есть сумма колонок.

А разве не о многом говорит тот факт, что для UNIX, по большому счету, проблемы вирусов не существует вообще? Вы можете встретить на прилавках магазинов книжки с названиями типа «Недокументированные возможности DOS» или «Секреты Windows», в то время как пользователям большинства UNIX-систем доступна не только подробнейшая документация, но и исходные тексты. Очевидными достоинствами являются открытость системы, надежность, защита, многозадачность и большое число приложений, в том числе поставляемых в комплекте с системой. Но широкие возможности системы предполагают высокие требования к аппаратуре, и вряд ли UNIX-компьютеры могут показать лучшее соотношение «цена/производительность» в диапазоне, характерном для персональных компьютеров. Впрочем, такое сравнение не совсем корректно. Дело в том, что задачи и, соответственно, философия UNIX в корне отличаются от философии MS-DOS. Да, UNIX занимает на диске заметно больше места, чем MS-DOS, и стоит дороже, но посмотрите, сколько возможностей она предоставляет и сколько приложений в себя включает. Например, в версию UNIX фирмы HP входят компиляторы языков C++, Pascal, Fortran, текстовый процессор TEX, XWindows и многое другое. А теперь прибавьте к «смешной» цене MS-DOS стоимость всех аналогичных пакетов и заодно подсчитайте, сколько все это многообразие занимает места на диске! И требования UNIX к аппаратным средствам ЭВМ, по современным меркам, даже заядлому пользователю DOS не покажутся такими уж сумасшедшими...

Есть ли альтернатива?

Основные системы, предлагаемые на рынке UNIX для ПК на базе Intel процессоров, рассмотрены в таблице 2. А если хотите познакомиться с рынком UNIX в России более подробно, то рекомендую посетить новую выставку «UNIX-Expo», которая пройдет в Москве с 7 по 10 февраля 1995 года.

Система Linux могла бы стать сильным конкурентом DOS и даже коммерческих UNIX-систем, если бы не ее хакерская направленность. Впрочем, я имею собственный опыт установки Linux дома и в школе — все прошло вполне гладко. Но, кроме того, Linux пока еще мало известна. Если вы выбрали Linux, то вам, наверное, будет небезынтересно оценить степень ее распространения, которую постоянно отслеживает «Harald Tveit Alvesstrand». Для этого достаточно отправить по электронной почте пустое письмо по адресу: LINUX-COUNTER@UNINETT.NO, заполнив поле Subject фразой: I use Linux at school (или home, work). В ответ придет полная форма для регистрации и текущее состояние счетчика. Заполнив форму, отошлите ее обратно (только не используя Reply), и вы получите новое текущее состояние счетчика с учетом вашего посильного вклада. Например, на пятницу 4 ноября было состояние, приведенное в таблицах 3 и 4.

Приглашаем к сотрудничеству!

И снова о молодежи, да и не только о ней. В нашей школе мной была организована неформальная группа любителей UNIX (Linux). Члены этой группы дают друг другу практические советы по использованию и по программированию в UNIX. Проводятся конкурсы программ, и, как заверили в редакции, лучшие задания и лучшие ответы на них будут опубликованы в рубрике журнала «Монитор», посвященной UNIX. Мы ни в коем случае не конкурируем с SUUG, так как являемся не коммерческой организацией пользователей и разработчиков, а организацией любителей. Для заинтересовавшихся привожу адрес:

KARLSON@SCH57.MCN.MSK.SU (Никита Кожекин) и PHIL@SCH57.MCN.MSK.SU (Крылов Филипп).

Вот, собственно, и все.
Think different, think UNIX! ■

Литература

1. **В. Водолазкий. Linux в России: грядет ли конкуренция MS-DOS?// Монитор. — 1994. — № 7.**
2. **В. Давыдов, А. Шелудяков. Операционные системы для современных распределительных вычислительных систем// Монитор. — 1994. — № 2.**
3. **А. Кондратюк, С. Ющенко, Л. Агафонов. Операционная система QNX// Монитор. — 1994. — № 1.**

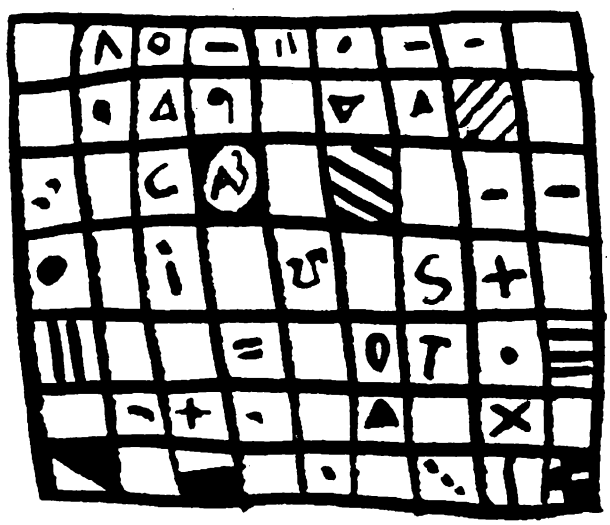
Таблица 4

Зарегистрированные пользователи LINUX

	Префикс e-mail	Страна	Используют сами	Знают о других пользователях	Всего	Население в млн. жителей	Процент пользователей Linux
1	is	Исландия	14	0	14	0,3	53,6
2	fi	Финляндия	232	0	232	5,0	46,4
3	no	Норвегия	147	0	147	4,3	34,2
4	se	Швеция	187	2	189	8,6	21,7
5	dk	Дания	110	3	113	5,1	21,6
6	nl	Нидерланды	314	2	316	14,9	21,1
7	au	Австралия	282	7	289	17,1	16,5
8	de	Германия	1172	14	1186	79,1	14,8
9	ca	Канада	383	22	405	26,6	14,4
10	us	США	3280	67	3347	249,6	13,1
11	at	Австрия	99	11	110	7,6	13,0
12	ch	Швейцария	80	0	80	6,7	11,9
13	si	Словения	19	1	20	2,0	9,7
14	gb	Великобритания	432	11	443	57,2	7,6
15	ie	Ирландия	25	0	25	3,5	7,1
16	be	Бельгия	68	506	574	9,9	6,9
17	nz	Новая Зеландия	20	2	22	3,4	5,9
18	ee	Эстония	9	4	13	1,6	5,6
19	lu	Люксембург	2	0	2	0,4	5,1
20	fr	Франция	279	14	293	56,2	5,0
21	sg	Сингапур	11	0	11	2,7	4,1
22	il	Израиль	22	0	22	6,3	3,5
23	hk	Гонконг	18	0	18	5,9	3,1
24	hu	Венгрия	25	0	25	10,5	2,4
25	cr	Коста-Рика	7	0	7	3,1	2,3
26	gr	Греция	18	0	18	10,1	1,8
27	it	Италия	95	0	95	57,7	1,6
28	jp	Япония	193	0	193	123,3	1,6
29	pt	Португалия	16	2	18	10,3	1,6
30	es	Испания	60	5	65	39,5	1,5
31	za	Южная Африка	41	0	41	30,2	1,4
32	cz	Чешская Республика	13	0	13	10,0	1,3
33	cl	Чили	17	0	17	13,5	1,3
34	hr	Хорватия	6	0	6	4,8	1,3
35	pa	Панама	3	0	3	2,6	1,2
36	tw	Тайвань	23	0	23	20,3	1,1
37	na	Намибия	1	0	1	1,5	0,7
38	sk	Словакия	3	1	4	5,4	0,6
39	pl	Польша	21	9	30	38,4	0,5
40	pr	Пуэрто-Рико	2	0	2	3,8	0,5
41	kr	Южная Корея	18	0	18	43,1	0,4
42	ro	Румыния	9	0	9	23,2	0,4
43	uy	Уругвай	1	0	1	3,2	0,3
44	lb	Ливан	1	0	1	3,6	0,3
45	ve	Венесуэла	5	0	5	20,1	0,2
46	ar	Аргентина	6	0	6	32,7	0,2
47	my	Малайзия	3	0	3	18,0	0,2
48	su	Россия (бывший СССР)	18	3	21	147,4	0,1
49	br	Бразилия	19	3	22	158,2	0,1
50	ua	Украина	6	0	6	51,9	0,1
51	bg	Болгария	1	0	1	8,8	0,1
52	by	Белоруссия	1	0	1	10,4	0,1
53	co	Колумбия	3	0	3	34,3	0,1
54	th	Таиланд	5	0	5	57,6	0,1
55	mx	Мексика	6	3	9	81,4	0,1
56	cs	Бывшая Чехословакия	1	0	1	15,7	0,1
57	ph	Филиппины	4	0	4	68,5	0,1
58	xe	Где-то в Европе	11	0	11	320,0	0,0
59	tr	Турция	2	0	2	60,9	0,0
60	pk	Пакистан	3	5	8	125,2	0,0
61	in	Индия	5	0	5	844,0	0,0
62	id	Индонезия	1	0	1	197,2	0,0
63	xw	Где-то в мире	10	0	10	5000,0	0,0
64	xx	Неизвестно...	3	19	22	5000,0	0,0

ГОСТ не прост..., а очень прост!

А. Винокуров



Судя по письмам, у читателей журнала есть весьма насущная потребность в криптографических алгоритмах различного рода и в реализующих их программных модулях. Однако многие слабо представляют себе криптографию вообще и существующие криптоалгоритмы в частности. Не все знают, что в России есть свой стандарт шифрования данных — ГОСТ 28147-89 [1], а многие из тех, кто слышал о нем, считают его чем-то очень сложным, доступным для понимания небольшой горстке профессионалов. Автор работает с этим алгоритмом три с половиной года, и за это время ему удалось создать неплохую, по его мнению, программную реализацию для процессоров семейства 80x86/88, приближающуюся по быстродействию к возможному оптимуму.

В настоящей статье вашему вниманию предлагается краткое описание алгоритма криптографического преобразования данных ГОСТ 28147-89, в дальнейшем немного фамильярно именуемого просто ГОСТом, и его программных реализаций.

С целью улучшить свою карму, а также чтобы не провоцировать коллег на воровство, автор предоставляет опубликованные в статье исходные тексты модулей в общественную собственность. Это значит, что вы можете свободно использовать их в своих разработках. В качестве компенсации за неполученную прибыль и ради удовлетворения своего тщеславия автор скромно, но убедительно просит вас упоминать его в документации на программные изделия, использующие созданный им код. При всем этом ваши декларации о том, что в вашем программном продукте используется российский стандарт шифрования, останутся пустым звуком, если у вас нет соответствующего сертификата от уполномоченной на то государственной организации. У автора такой бумаги нет, поэтому вам придется поверить ему на слово, либо не пользоваться публикуемыми кодами.

Настоящая статья состоит из нескольких частей. Первая часть введет вас в курс проблемы, вторая содержит краткое описание алгоритма шифрования, третья посвящена стандартным режимам

шифрования, в четвертой описаны маленькие хитрости, позволившие повысить быстродействие реализации, пятая содержит описание кода, в шестой анализируются достигнутые параметры.

Почему ГОСТ ?

Думаю, никому сейчас не надо объяснять важность информации в современном мире. Теперь обладание определенным цифровым кодом может открыть его владельцу доступ к значительным материальным ценностям и услугам. Ясно, что в этих условиях информация становится потенциальным объектом различных злоупотреблений, что, в свою очередь, приводит к насущной потребности в средствах противодействия этому.

Одна из классических и, очевидно, древнейшая задача такого рода — это защита сообщений от несанкционированного ознакомления с их содержимым. Задача настолько актуальна, что практически в каждом уважающем себя государстве есть свой стандарт шифрования. Спрос определяет предложение — надо сказать, что недостатка в предложении алгоритмов и программ подобного типа нет (см., например, [2] или любой аналогичный каталог). Вам наперебой предлагают «новые» «сверхбыстрые» и «сверхкриптостойкие» программы. При ближайшем рассмотрении, однако, многие из них по потенциальным результатам своего применения сильно напоминают краситель для волос «Титаник», купленный небезызвестным Ипполитом Матвеевичем Воробьяниновым по случаю у аптекаря. Все это вызывает удивление, ведь в России есть свой стандарт шифрования — ГОСТ 28147-89, принятый, как явствует из его кода, еще в СССР в 1989 году.

До сравнительно недавнего времени он имел гриф, однако сейчас он открыт, по крайней мере, описание ГОСТа приводится в открытом издании [3]. Конечно, как справедливо замечают его критики, алгоритм слегка медлителен, но все же не настолько, чтобы отказаться от его применения — ведь на другой чаше весов высочайший интеллект специалистов-разработчиков и мощь стоявшей за этой разработкой организации (догадайтесь, какой). Криптостойкость, заложенная в этот алгоритм, удовлетворит самого взыскательного хранителя секретов!

О потенциальных возможностях реализаций этого алгоритма знают немногие — забавно, но даже профессионалы зачастую попадают впросак, когда делают заявления типа «.... за одну секунду на персональном компьютере может быть обработано лишь несколько десятков (максимально сотен) байтов данных...» — уважаемые авторы книжки [3] ошиблись раз, эдак, в сто! Кстати, почти во всех случаях вина за медлительность лежит не на алгоритме, а на его реализации. Характерный пример: замена модулей реализации ГОСТа, составленных на языке С, ассемблерными модулями, предложенными в этой статье, увеличила интегральную скорость работы программы шифрования на ПЭВМ типа IBM PC приблизительно в 8 раз.

Основные понятия

Официально ГОСТ называется «Алгоритм криптографического преобразования данных ГОСТ 28147-89» — это несколько шире, чем просто зашифровывание или расшифровка данных. Все режимы криптопреобразований данных, согласно ГОСТ, как на трех китах, базируются на трех циклах алгоритма, так и называемых автором для краткости базовыми:

- цикл зашифровывания (32-З);
- цикл расшифровки (32-Р);
- цикл выработки имитоприставки (16-З);

Смысл кода, приведенного в скобках, станет понятен читателю несколько позже. В свою очередь, каждый из этих циклов представляет собой вариацию на тему одного-единственного основного шага криптопреобразования. Таким образом, чтобы разобраться в ГОСТе, надо понять следующие вещи:

а) что такое основной шаг криптопреобразования;
б) как из основных шагов складываются базовые циклы;
в) как из трех базовых циклов складываются все практические алгоритмы ГОСТа.

Прежде чем перейти к изучению этих вопросов, поговорим о дополнительной информации, используемой ГОСТом, — именно ее секретность обеспечивает секретность шифрованного сообщения. Эта информация представляет собой 2 массива данных — ключ и таблицу замен, приведем их характеристики.

1. *Ключ* — это массив из 8 32-битовых элементов, обозначаемых в дальнейшем X_i, где i изменяется от 0 до 7. Таким образом, размер ключа составляет 32×8=256 битов, или 32 байта.
2. *Таблица замен* — двумерная таблица — набор из 8 одномерных массивов (узлов замен), каждый из которых содержит 16 различных 4-битовых чисел (от 0 до 15) в произвольном порядке. Обозначим $K_m(l)$ значение l-го элемента в m-ом узле замен. При этом m изменяется в пределах 0...7, а l — в пределах 0...15. Таким образом, общий объем таблицы замен равен 8 узлов ×16 элементов ×4 бита/элемент = 512 битов = 64 байта.

Рассмотрим основной шаг криптопреобразования, структура его алгоритма представлена на рисунке 1. На входе шага заданы два 32-битовых элемента данных — N_1, N_2, с этими элементами выполняются следующие манипуляции:

1) добавление к N_1 элемента ключа — сложение по модулю 2^32.
2) поблочная замена результата по 4 бита по таблице замен;
3) циклический сдвиг результата на 11 битов влево;
4) побитовое сложение результата по модулю 2 с элементом N_2;
5) перестановка элементов: N_2 <- старое N_1, N_1 <- результат;

После этого новые элементы N_1, N_2 выдаются в качестве результата шага. Так как в основном шаге используется только один элемент ключа, еще одним параметром шага является номер этого элемента.

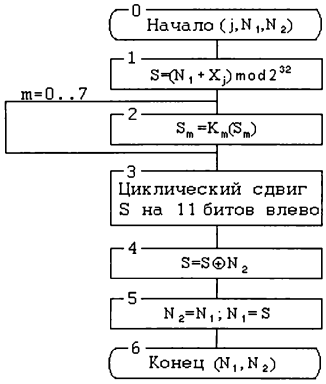


Рис. 1. Схема алгоритма основного шага криптопреобразования

Рассмотрим базовые циклы криптоалгоритма ГОСТа. Они отличаются друг от друга только числом повторений основного шага и порядком просмотра элементов ключа. В обозначении цикла nn-X первый элемент (nn) — это число повторений основного шага, а второй кодирует порядок просмотра элементов ключа (буква З — порядок зашифровывания или Р — расшифровки). Кроме того, в конце циклов шифрования предусмотрена дополнительная перестановка элементов. Приведем порядок использования элементов ключа для трех базовых циклов:

1) цикл зашифровывания (32-З) — 3 раза вперед, 1 раз назад:
0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,7,6,5,4,3,2,1,0
2) цикл расшифровки (32-Р) — 1 раз вперед, 3 раза назад:
0,1,2,3,4,5,6,7,7,6,5,4,3,2,1,0,7,6,5,4,3,2,1,0,7,6,5,4,3,2,1,0
3) цикл выработки имитоприставки (16-З) — 2 раза вперед:
0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7

Блок-схемы базовых циклов приведены на рисунках 2, 3, 4. Каждый из циклов получает на входе 2 32-битовых слова и после серии основных шагов выдает в качестве результата также 2 32-битовых слова.

Основные режимы шифрования

ГОСТ 28147-89 предусматривает три режима шифрования данных:

1) простая замена;
2) гаммирование;
3) гаммирование с обратной связью;

и дополнительный режим

4) выработка имитоприставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита — именно поэтому ГОСТ относится к блочным шифрам. Кратко опишем основные режимы шифрования.

Простая замена

Зашифровывание заключается в применении цикла 32-З к блокам открытого текста, расшифровка — в применении цикла 32-Р к блокам шифротекста. Это наиболее простой режим шифрования, и он имеет следующие недостатки:

- с точки зрения стойкости шифра, одинаковые блоки исходных данных дают одинаковые блоки шифротекста; криптологи говорят, что это очень плохо;
- с точки зрения удобства применения, если длина массива информации не кратна 8 байтам, то возникают 2 небольшие проблемы:
  - чем и как дополнять последний блок до полных 8 байтов;
  - после зашифровывания неполного блока в нем все 8 байтов станут значащими, то есть вместе с шифротекстом надо хранить количество байтов в последнем блоке исходного текста.

ГОСТ ограничивает возможные случаи применения простой замены шифрованием ключевой информации (ключи и таблицы замен).

Гаммирование

Этот режим заключается в наложении на открытые данные гаммы с помощью побитовой функции «исключающее или». Зашифровывание и расшифровка в этом режиме не отличаются друг от друга.

$T^{ш}_i = T^{o}_i \oplus Гi$

$T^{o}_i = T^{ш}_i \oplus Г_i$

для i=1...K (K-число блоков данных в шифруемом массиве).

Здесь T^{o}_i, $T^{ш}_i$, $Г_i$ — 64-битовые блоки открытого текста, шифротекста, гаммы соответственно. Блоки гаммы получаются зашифровыванием в режиме простой замены некоторой последовательности 64-битовых блоков, вырабатываемых датчиком псевдослучайных чисел. От этого датчика не требуется обеспечения никаких статистических характеристик выходной последовательности, а нужен лишь максимальный период повторения данных. Схема алгоритма шифрования в режиме гаммирования приведена на рисунке 5. В качестве входного параметра алгоритма шифрования в этом режиме используется 64-битовый блок данных S, именуемый синхропосылкой, или начальным заполнением. Константы C_1, C_2 из блока генерации гаммы имеют следующие значения в шеснадцатеричном представлении:

C_1=01010101h

C_2=01010104h

Гаммирование с обратной связью

Данный режим похож на режим гаммирования и отличается от него только тем, что для выработки блока гаммы для шифрования следующего блока данных используется блок шифротекста, полученный на преды-

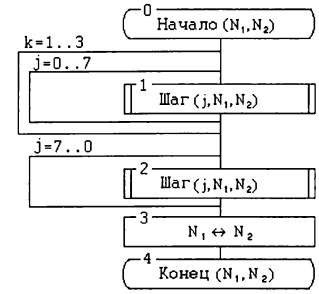


Рис. 2. Схема алгоритма базового цикла зашифровывания 32-З

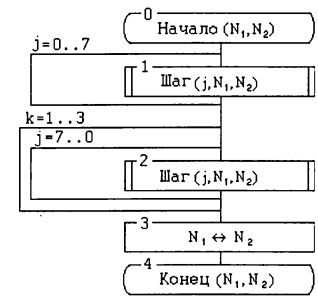


Рис. 3. Схема алгоритма базового цикла расшифровки 32-Р

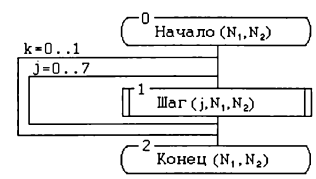


Рис. 4. Схема алгоритма базового цикла выработки имитоприставки 16-З

дущем шаге. Этим достигается зацепление блоков — каждый блок при шифровании зависит от всех предыдущих. Какое это имеет значение, поясним на примере: пусть почтальон-злоумышленник (лицо, имеющее полный доступ к шифрованным данным, но не владеющее ключом и не способное расшифровать сообщение) узнал, что в передаваемом сообщении 13-й байт от начала является старшим разрядом числа, например суммы некоторого контракта. Пусть также ему известно значение этой цифры, допустим, это будет 1. Тогда он элементарно может заменить ее на любую другую цифру, например на 2, не расшифровывая сообщения:

$A_{13} = A_{13} \oplus$ '1' $\oplus$ '2'

Если бы данные шифровались в режиме простой замены, это привело бы к порче всего блока — после расшифровки мы получили бы совершенно случайную комбинацию символов на месте с 9-го по 16-й байты нашего текста. Если зашифровывание выполнено в режиме гаммирования, то после расшифровки мы получим текст, в котором в 13-й позиции вместо верной цифры 1 стоит цифра 2 и больше никаких изменений нет! Очевидно, не надо объяснять, насколько потенциально опасной может быть такая подмена. Если же зашифровывание выполнено в режиме гаммирования с обратной связью, то после расшифровки мы получим в блоках данных до испорченного включительно тот же результат и при гаммировании, но все последующие блоки окажутся запорченными, то есть после расшифровки их содержимое окажется бессмысленной комбинацией расширенных кодов ASCII. Отсюда становится ясно, что шифрование в режиме гаммирования с обратной связью делает очевидным нарушение целостности данных, если только изменения не внесены лишь в последний блок данных шифротекста. Схемы алгоритмов шифрования в режиме гаммирования с обратной связью приведены на рисунках 6, 7.

Выработка имитоприставки к массиву данных

Имитоприставка — это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. Цель использования имитоприставки — обнаружение всех случайных или преднамеренных изменений в массиве информации. Проблема, описанная в предыдущем пункте, может быть успешно решена с помощью добавления к шифрованным данным имитоприставки. Для потенциального злоумышленника две следующие задачи, если он не владеет секретным ключом, практически неразрешимы:

- вычисление имитоприставки для заданного открытого массива информации;
- подбор открытых данных под заданную имитоприставку;

Схема алгоритма выработки имитоприставки приведена на рисунке 8. В качестве имитоприставки берется не весь блок, полученный на выходе последнего цикла 16-3, а только его часть. Как правило, это 32 младших бита блока, то есть N_1.

Замечания по реализации

Можно, конечно, составить программные модули, точно реализующие приведенные в настоящей статье схемы алгоритмов, при этом мы получим довольно неплохую и очень компактную реализацию ГОСТа. Но профессиональный подход к делу требует большего. Рассмотрим возможности получения максимально быстрых и по возможности компактных кодов реализации ГОСТа.

Если внимательно посмотрим на схему основного шага, мы увидим, что в нем 8 раз выполняется подстановка 4-битовых данных. Однако процессор, для которого мы делаем реализацию алгоритма (80x86), имеет удобную команду перекодировки 8-битовых данных (xlat). Использование данной команды дает нам возможность за один шаг выполнять две подстановки и избавляет от необходимости выделять полубайты из двойных слов. Этим достигается существенное увеличение быстродействия кода, однако достаточно дорогой ценой — для реализации данного подхода мы должны преобразовать таблицу замен: каждая пара узлов замен, требующая для своего представления 2×16×4 бита = 16 байтов, заменяется одним 256-байтовым массивом, говоря языком математики, прямым произведением этой пары узлов замен. При таком подходе таблица замен занимает 256×4=1024 байта =1 кб. Автор считает такую плату за быстродейсвие вполне приемлемой.

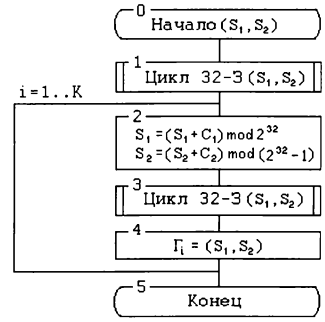


Рис. 5. Схема алгоритма выработки гаммы для шифрования данных

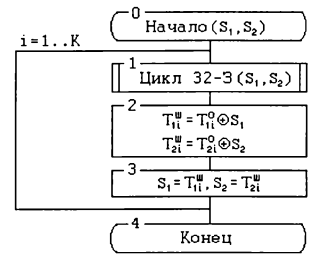


Рис. 6. Схема алгоритма зашифровывания блока данных в режиме гаммирования с обратной связью

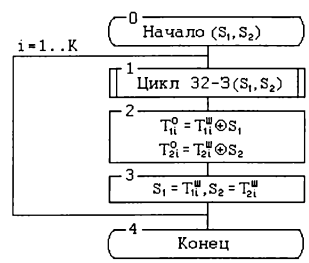


Рис. 7. Схема алгоритма расшифровки данных в режиме гаммирования с обратной связью

Далее заметим, что после перекодировки по таблице замен необходимо выполнить циклический сдвиг 32-битового слова на 11 разрядов влево. Так как процессоры семейства 80x86 с перируют 16-разрядными данными, реализация такого вращения может оказаться несколько громоздкой. Однако часть проблемы может быть решена на предыдущем шаге: надлежащим порядком перестановки байтов для их перекодировки в регистре AL можно достичь и вращения данных на 1 байт, то есть на 8 битов влево. Тогда на следующем шаге останется только «довернуть» данные на 3 бита.

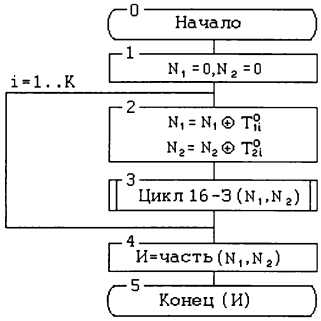


Рис. 8. Схема алгоритма выработки имитоприставки для блока данных

Далее заметим, что все три базовых цикла отличаются друг от друга следующим образом:

- циклы шифрования повторяют основной шаг 32 раза, тогда как цикл выработки имитоприставки — 16 раз;
- порядок просмотра элементов ключа различный для различных циклов;
- в конце циклов шифрования необходимо выполнить перестановку полученных двойных слов (рис. 2, 3, блок 3), в цикле выработки имитоприставки этого не требуется.

Очевидно, эти различия столь незначительны, что все три базовых цикла могут быть реализованы в одном программном модуле. При этом для повышения эффективности кода необходимо развернуть ключ в памяти машины до массива, который бы в ходе базовых циклов просматривался лишь однократно. На первый взгляд покажется, что такой ключ должен быть сделан для каждого цикла, что в сумме потребует десятикратного (4+4+2=10) увеличения размера ключа. Однако, если хорошо подумать, можно понять, что достаточно ограничиться лишь шестикратным повторением — при этом ключевой массив будет содержать элементы ключа три раза в прямом и три раза в обратном порядке. Схема использования различными циклами элементов ключа приведена на рисунке 9. Более того, если программа не использует цикл 32-Р, как приведенная в статье программа cryptor, можно ограничиться только 4-кратным увеличением размера ключа. Отсюда становится ясно, что первые два различия между циклами реализуются передачей в модуль в качестве параметра числа повторений основного шага и адреса первого используемого элемента ключа. Третье различие в базовых циклах может быть учтено на этапе использования полученных в результате работы цикла данных.

С учетом сделанных замечаний автору удалось создать реализацию универсального базового цикла ГОСТа с минимально возможным обращением к оперативной памяти машины. Думается, что предлагаемая реализация базового цикла близка к возможному оптимуму для 16-битовых процессоров семейства 80x86. Использование кода для 32-битовых процессоров этого семейства позволяет дополнительно значительно увеличить быстродействие, и такая реализация имеется у автора, однако ограниченный объем статьи не позволяют привести соответствующие листинги, как-нибудь в другой раз.

Листинги

В качестве приложения к настоящей статье приводятся листинги 1-11 программных модулей, реализующих

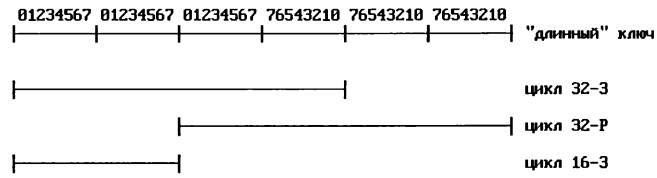


Рис. 9. Использование различными базовыми циклами фрагментов расширенного ключа

все режимы работы ГОСТа. Перечень листингов дан в таблице 1. При разработке программ автор использовал системы программирования Borland C++ 2.0 и 3.1 и соответствующие ассемблеры — TASM 2.5 и TASM 3.1. Ради универсальности все подпрограммы сделаны так, что принимают «дальние» указатели на используемые блоки данных, это также облегчает их стыковку с модулями на Pascal. Если же сделать свой код для каждой модели памяти, можно выиграть еще несколько байтов памяти и тактов времени. Имя кодового сегмента (gost), заданное в директиве .model в ассемблерных модулях,

Таблица 1

Перечень прилагаемых листингов

Листинг	Файл	Назначение модуля
1	gost.asm	Универсальный шаг криптопреобразования по ГОСТ 28147-89
2	simple.asm	Шифрование блока данных в режиме простой замены
3	gamma.asm	Шифрование блока данных в режиме гаммирования
4	gammale.asm	Зашифровывание блока данных в режиме гаммирования с обратной связью
5	gammald.asm	Расшифровка блока данных в режиме гаммирования с обратной связью
6	imito.asm	Выработка имитоприставки для блока данных
7	expcht.c	Построение расширенной таблицы замен
8	expkey31.c	Построение расширенного ключа по схеме 3+1; (три раза в прямом направлении, 1 в обратном)
9	gost.h	Описание данных и прототипы функций для предложенной реализации
10	cryptor.c	Исходный тест программы шифрования
11	cryptor.mak	Проект построения программы шифрования

необходимо для сбора всех процедур криптопреобразования в один сегмент при использовании моделей памяти с несколькими сегментами кода. Если вас раздражает вызванное этим предупреждение, выдаваемое ассемблером, удалите имя сегмента (gost) из исходных текстов.

Для иллюстрации использования предложенных кодов приведен исходный текст программы шифрования файлов на языке С. Для построения программы шифрования задайте следующую команду:

```
make -fcryptor.mak
```

Для зашифровывания файла наберите следующую команду:

```
cryptor e /ключи... источник [приемник]
```

Расшифровку файла осуществляйте с помощью следующей команды:

```
cryptor d /ключи... источник [приемник]
```

Здесь:

источник — файл, подлежащий зашифровыванию или расшифровке;

приемник — файл, куда должен быть записан результат шифрования, если файл-приемник не задан, он записывается на место источника. Если приемник задан, файла с таким именем не должно существовать — он будет создан программой, иначе фиксируется ошибка;

/ключ — один или несколько ключей из следующего списка:

/b — шифровать с обратной связью; если ключ не задан, проводится шифрование в режиме гаммирования, если же он задан, реализуется гаммирование с обратной связью;

/d — после завершения шифрования файл-источник будет удален;

/sXXXX — задает синхропосылку для шифрования, здесь *XXXX* — синхропосылка — 8-разрядное шестнадцатеричное целое число без знака и маркеров, то есть от 1 до 16 шестнадцатеричных цифр (под маркером константы понимаем код ее основания — h в конце константы в ассемблере и 0x в начале константы в С); по умолчанию XXXX=0; этот ключ позволяет использовать синхропосылку как своеобразный пароль при зашифровывании.

/kXXXX — задает файл с ключом шифрования; XXXX — стандартное для MS-DOS имя файла; если этот ключ не задан, будет использован тривиальный ключ из одних нулей; файл должен иметь длину 32 байта и содержать обычный ключ или 128 байтов и содержать уже развернутый ключ;

/cXXXX — задает файл с таблицей замен; XXXX — стандартное для MS-DOS имя файла; если этот ключ не задан, будет использована тривиальная таблица замен, произойдет замена любого значения на него же; формат таблицы замен должен быть следующим: 8 узлов замен по 16 элементов каждый, причем каждый элемент замены хранится в целом байте, а не в полубайте; таким образом, размер файла с таблицей замен должен быть равен 16×8=128 байтам. Допустимо, чтобы файл содержал уже развернутую таблицу замен, в этом случае его длина должна быть 1 кб;

/i[XXXX] — выработать имитоприставку к данным; при расшифровке может быть задана имитоприставка, выработанная при зашифровывании, XXXX задается в виде 8-значного шестнадцатеричного целого без знака и маркеров; в этом случае на экран будет выведено сообщение о нарушении целостности данных, если при расшифровке получено другое значение имитоприставки; если ключ /i задан без параметра XXXX, имитоприставка будет вычислена и выведена на дисплей; если ключ не задан, имитоприставка не вырабатывается.

Ключ шифрования и таблица замен, используемые в программе по умолчанию, являются тривиальными, то есть их применение не меняет соответствующие данные и полученный шифротекст легко поддается криптоанализу. Для получения действительно стойкого шифра используйте «хорошую» ключевую информацию из внешних файлов или внесите ее в исходный текст программы cryptor в качестве инициаторов массивов Key (ключ) и chtab (таблица замен), учтите только, что эти массивы содержат расширенные ключевые данные.

Чего же мы достигли ?

Полученная программа, транслированная с размером обменного буфера 8 кб, была протестирована на двух ПЭВМ:

1) «Искра 1031», процессор — аналог Intel 8088, 8 МГц;

2) IBM AT совместимая машина, процессор Intel 386-SX, 16 МГц;

В качестве теста программа зашифровывала и расшифровывала файл размером 1 Мб в режиме с выработкой имитоприставки и без нее. В одинаковых режимах зашифровывание и расшифровка дали идентичные временные характеристики, что вполне закономерно. Результаты испытаний приведены в таблице 2. Попытаемся дать по ним оценку быстродействия кодов. Учтем, что цикл выработки имитоприставки вдвое короче и вдвое быстрее цикла шифрования. Разница времени, затраченного на шифрование с выработкой имитоприставки и без нее, характеризует скорость ее выработки, скорость шифрования вдвое меньше. Если руководствоваться этими принципами, получим результаты, представленные в той же таблице 2. Согласно им, производительность шифрования на CPU Intel 8088 8 Мгц составляет около 10 кб/с, а на CPU Intel 80386-SX 16 МГц — 28 кб/с. Это не так много, но вполне достаточно для очень многих применений.

Что же является альтернативой? Малым предприятием «Анкад» производится аппаратная плата шифрования данных согласно ГОСТ 28147-89 «Криптон-3». В ее паспортных данных заявлена производительность шифрования 50 кб/с. Она, наверное, и обеспечивается, но на

Таблица 2

Время шифрования файла размером 1Мб программой cryptor

Выработка имитоприставки	Тип ПЭВМ и процессора	
	Искра 1031(аналог Intel 8088 8Мгц)	AT-386(Intel 80386-SX 16Мгц)
Есть	210 с	65 с
Нет	160 с	48 с
Оценка быстродействия	10 кб/с	28 кб/с

уровне «железа». Реальная производительность при обращении к плате через ее ROM-BIOS ниже за счет того, что некоторое, и притом весьма значительное время выполняются модули из BIOS платы. Автором разработано несколько программ, шифрующих данные как чисто программно, так и с помощью платы «Криптон-3». Различие по интегральному быстродействию между ними колеблется от 1,9 до 2,1 раза для разных машин при 16-битовой реализации криптоалгоритма, приведенной в данной статье, и от 1,4 до 1,6 раза при 32-битовой реализации, естественно, в пользу аппаратного шифрования. С учетом прямо-таки астрономической цены платы (около 300 тыс. руб. в начале лета 1994 года) не покажется, что это много. Если же пользователя не устроят такие скорости, то выход один — применять другие, более быстрые алгоритмы. Возможно, о некоторых из них удастся рассказать на страницах журнала в будущем.

Заключительные замечания

Если читатель внимательно изучит оригинал ГОСТ 28147-89, он заметит, что схемы некоторых алгоритмов, приведенных в данной статье, несколько отличаются от их прототипов в оригинальном тексте стандарта. В свое оправдание автор скажет, что он пришел к нынешним схемам после достаточно длительного осмысления алгоритма и попыток упростить его понимание (но ни в коем случае не сам алгоритм!). Для сомневающихся задача: доказать или опровергнуть функциональную эквивалентность приведенных в статье и в оригинале ГОСТа схем алгоритмов. Указание к решению: применить метод эквивалентных преобразований блок-схем.

Приведенная в качестве примера программа шифрования файлов cryptor.c носит иллюстративный характер и не является верхом совершенства ни в чем, кроме быстродействия реализации ГОСТа, хотя многих может научить хорошему стилю программирования на С. Что касается надежности, то использование самых прекрасных алгоритмов шифрования само по себе не может сделать вашу систему надежной — многое, очень многое зависит от протоколов криптосистемы. Так, в приведенной программе шифрования файлов хранение ключевой информации (ключ, таблица замен) на дисках в открытом виде делает систему, которая была бы на ней реализована, потенциально ненадежной.

Кроме того, программа составлена в сжатые сроки специально для публикации в данной статье, поэтому вполне вероятно, что она содержит некоторые ошибки, несмотря на предпринятые усилия все их «отловить», автор очень сильно извиняется за них.

Среди приведенных листингов отсутствует текст подпрограммы построения развернутого ключа по схеме 3+3. Для читателей не составит труда получить ее элементарной доработкой процедуры ExpKey31, для построения же приведенного в статье примера эта подпрограмма не нужна.

На этом автор заканчивает свою статью и в качестве совсем уж последнего замечания делится с вами своими планами опубликовать в одном из следующих номеров журнала еще одну статью, посвященную различным вариациям на тему ГОСТа,— его упрощенным модификациям и нестандартным применениям для различных нужд — выработки парольной и ключевой информации, контроля неизменности массивов данных и даже цифровой подписи. ■

Замечания вдогонку

Моя статья о реализации криптографического алгоритма ГОСТ 28147-89 была написана в мае-июне 1994 года, однако по различным причинам ее публикация задержалась. За это время многое успело произойти — в частности, была создана BBS факультета защиты информации РГГУ (т. 388-40-66 в Москве, на конец осени 1994 года работает в дневное время по рабочим дням, нерегулярно). Через эту BBS можно задать автору вопросы и получить ответы на них.

Кроме того, через нее можно получить следующие материалы:

- дополнения и уточнения к опубликованным статьям автора;
- исходные коды модулей;
- перечень распространяемых программных средств с условиями их распространения;
- FreeWare и ShareWare программы.

Абоненты сети Relcom могут присылать мне свои замечания по адресу Andy@RGGU.msk.su

А. Винокуров

Литература

1. **ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.**
2. **Защита информации. Каталог отечественных средств, систем, услуг. Первый выпуск. — М: АО Рескрипт, 1992.**
3. **А.В. Спесивцев и др. Защита информации в персональных ЭВМ. — М: Радио и связь, 1992.**

Листинг 1

```
;-------------------------------------------------------
; Реализация универсального цикла шифрования алгоритма
; криптографического преобразования ГОСТ 28147-89.
; Copyright (C) 1992-1994 Винокуров А.Ю., г. Москва.
;-------------------------------------------------------
;>Параметры  при  вызове:
;   (DX,AX)=N1, (BP,DI)=N2;
;   ES:SI - адрес ключа;
;   DS:BX - адрес таблицы замен;
;   CX    - число основных шагов
;>Результаты на выходе:
;   (BP,DI)=N1, (DX,AX)=N2 для циклов 32-3,32-P;
;   (DX,AX)=N1, (BP,DI)=N2 для цикла  16-3;
;>Используемые регистры:  ВСЕ
;>Замечания:
;  В конце выполнения содержимое регистров следующее:
;  BX-указатель таблицы замен - то же, что в начале
;  SI-указатель ключа - указывает на первый байт за
;     ключом - это на 32*4=128 больше первоначального
;     значения  SI для циклов шифрования и на 16*4=64
;     больше первоначального значения для цикла выра-
;     ботки имитоприставки
;  CX=0
;  Содержимое сегментных регистров не изменяется.
;-------------------------------------------------------
;
  .model  small C
        .code $gost
gost    proc    near
        public  gost
; Внутренний  цикл  работы  П/П
; 1. Начало  цикла  и  сохранение  старого  N1
iloop:  push    DX
        push    AX
; 2. Добавление  к  N1  ключа  по  модулю  2^32
        add     AX,ES:[SI]
        adc     DX,ES:[SI+2]
        add     SI,4
; 3. Поблочная замена в S со сдвигом на 8 битов влево!
        xlat            ; перекодируем AL
        xchg    AH,AL   ;  и меняем его с AH
        inc     BH      ;   продвигаем указатель замены
        xlat            ; перекодируем AH
        xchg    DL,AL   ;  и меняем его с DL
        inc     BH      ;   продвигаем указатель замены
        xlat            ; перекодируем DL
        xchg    DH,AL   ;  и меняем его с DH
        inc     BH      ;   продвигаем указатель замены
        xlat            ;  перекодируем DH
        sub     BH,3    ; BX <- его прежнее значение
; 4. Вращение  S  еще  на  3  бита  влево
REPT 3
        shl     AX,1
        rcl     DX,1
        adc     AX,0
ENDM
; 5. Вычисление  новых  значений  N1,N2
        xor     DX,BP
        xor     AX,DI
        pop     DI
        pop     BP
; Завершение  внутреннего  цикла
  loop    iloop
        ret
gost    endp
        end
```

■

Листинг 2

```
;-------------------------------------------------------
; Шифрование массива данных в режиме простой замены
; Copyright (C) 1992-1994 Винокуров А.Ю., г. Москва.
;-------------------------------------------------------
; Параметры  при  вызове  передаются  в  стеке :
; Смещение  Размер    П  А  Р  А  М  Е  Т  Р
; NEAR FAR
;  2    4      4      FAR-адрес ключа
;  6    8      4      FAR-адрес блока данных
; 10   12      4      FAR-адрес таблицы замен
; 14   16      2      Количество шифруемых блоков
;-------------------------------------------------------
; При выходе: адрес шифруемой области скорректирован.
; Счетчик длины = 0
;-------------------------------------------------------
; Зашифровывание или расшифровка задается передачей
; соответствующего адреса ключа - порядок элементов
; в ключах  зашифровывания  и  расшифровки взаимно
; обратный.
;-------------------------------------------------------
;
        .model  small C
        .code $gost
        extrn   gost:near
simple  proc
        public  simple
; вычисляем смещение параметров в стеке
        a=8             ; 4 регистра в стеке
IF      (TYPE simple) EQ OFFFEh
        a=a+2           ; если simple - far-процедура
ENDIF
; Задаем  смещение  аргументов  в  стеке
;
key=    2+a             ; смещение адреса ключа
syn=    6+a             ; смещение адреса данных
chg=    10+a            ; смещение адреса таблицы замен
len=    14+a            ; смещение счетчика блоков
;
; Сохранение регистров в соответствии с соглашением Turbo C
  push  BP
  push  DS  ; это не всегда обязательно
  push  SI
  push  DI
; Настройка рабочих регистров
        mov     BP,SP   ; стековый кадр
        lds     BX,chg[BP]  ; таблица замен
; Загрузка  блока  в  регистры
circle: les     SI,syn[BP]  ; загрузка адреса С-посылки
        mov     AX,ES:[SI] ; AX <- младшее слово N1
        mov     DX,ES:2[SI]; DX <- старшее слово N1
        mov     DI,ES:4[SI]; DI <- младшее слово N2
        mov     CX,ES:6[SI]; CX <- старшее слово N2
; Готовим регистры и вызываем простую замену
        les     SI,key[BP]
        mov     BP,CX
        mov     CX,32
        call    gost
  mov     CX,BP
  mov     BP,SP

; Запись  результата  на  место
        les     SI,syn[BP]      ; загрузка  адреса
синхропосылки
        mov     ES:[SI],DI          ; з а н о с и м
        mov     ES:2[SI],CX         ;   р е з у л ь т а т
        mov     ES:4[SI],AX         ;       н а     е г о
        mov     ES:6[SI],DX         ;           м е с т о !!!
; Проверка на конец цикла
        dec     WORD PTR len[BP]; коррекция счетчика
        jnz     circle          ; циклимся

; Восстановление регистров
  pop   DI
  pop   SI
  pop   DS
  pop   BP
  ret
simple  endp
  end
```

■

Листинг 3

```
;-------------------------------------------------------
; Шифрование массива данных в режиме гаммирования
; согласно криптоалгоритму ГОСТ 28147-89
; Copyright (C) 1992-1994 Винокуров А.Ю.,  г. Москва.
;-------------------------------------------------------
; Параметры  при  вызове  передаются  в  стеке :
; Смещение Размер    П  А  Р  А  М  Е  Т  Р
;NEAR FAR
;  2   4      4     FAR-адрес ключа
;  6   8      4     FAR-адрес синхропосылки (S1,S2)
; 10  12      4     FAR-адрес таблицы замен
; 14  16      4     FAR-адрес области назначения
; 18  20      2     число шифруемых блоков
;-------------------------------------------------------
; При выходе: адрес  области  назначения  и  длина
;             содержат скорректированные параметры
;-------------------------------------------------------
;
   .model  small C
   .code $gost
   extrn   gost:near
gamma   proc
   public  gamma
; вычисляем смещение параметров в стеке
        a=8             ; 4 регистра в стеке
IF      (.TYPE gamma) EQ OFFFEh
   a=a+2            ; если gamma - far-процедура
ENDIF
; Задаем  смещение  аргументов  в  стеке
;
key=    2+a
syn=    6+a
chg=    10+a
con=    14+a
tar=    14+a
len=    18+a
;
; Сохранение регистров согласно соглашениям Turbo C
   push    BP
   push    DS          ; это нужно не всегда!
   push    SI
   push    DI
; настройка регистров
        mov     BP,SP       ; стековый кадр
        lds     BX,chg[BP]  ; DS:BX <<- адрес таблицы замен
; Добавление  констант
circle: lds     SI,syn[BP] ; адрес    синхропопылки
   mov     AX,ES:[SI]  ; AX <- младшее слово S1
   mov     DX,ES:2[SI] ; DX <- старшее слово S1
   mov     DI,ES:4[SI] ; DI <- младшее слово S2
   mov     CX,ES:6[SI] ; CX <- старшее слово S2
   add     AX,CS:C1low ; изменение S1:
   adc     DX,CS:C1high;    S1=(S1+C1) mod 2^32
   stc                 ; для добавления 1 к N2
   adc     DI,CS:C2low ; изменение S2:
   adc     CX,CS:C2high;  S2=(S2+C2) mod (2^32-1)
   cmc                 ; если не было переноса,то
   sbb     DI,0        ;  не стоило добавлять 1,
   sbb     CX,0        ;   поэтому S2=S2-1
   mov     ES:[SI],AX  ; заносим новые
   mov     ES:2[SI],DX ;    значения S1,S2
   mov     ES:4[SI],DI ;       на   их
   mov     ES:6[SI],CX ;          место !!!
; Готовим регистры для вызова цикла 32-3;
   les     SI,key[BP]  ; ES:SI <- адрес ключа
   mov     BP,CX       ; BP <- старшее  слово  S2
        mov     CX,32       ; CX <- число основных шагов
   call    gost        ; шаг  простой  замены
   mov     CX,BP       ; CX <- старшее  слово  S2
   mov     BP,SP       ; BP <- стековый  кадр
; Использование  сгенерированного  блока  гаммы
   les     SI,tar[BP]  ; DS:SI<-адрес  назначения
   xor     ES:[SI],DI  ; Гаммируем
   xor     ES:2[SI],CX    ;    8-байтовый
   xor     ES:4[SI],AX    ;       блок
   xor     ES:6[SI],DX    ;      данных
   add     word ptr tar[BP],8; коррекция указателя
   dec     word ptr len[BP]  ; коррекция счетчика
   jnz     circle      ; на генерацию нового блока
; Восстановление регистров из стека
   pop     DI
   pop     SI
   pop     DS
   pop     BP
   ret
; Константы C1, C2 - лишние 4 байта ради универсальности
C1low   dw   0101h
C1high  dw   0101h
C2low   dw   0104h
C2high  dw   0101h
gamma   endp
   end
```

Листинг 4

```
;-------------------------------------------------------
; Зашифровывание массива данных в режиме гаммирования с
; обратной связью согласно алгоритму ГОСТ 28147-89
; Copyright (C) 1992-1994 Винокуров А.Ю.,  г. Москва.
;-------------------------------------------------------
; Параметры  при  вызове  передаются  в  стеке :
; Смещение Размер    П  А  Р  А  М  Е  Т  Р
;NEAR FAR
;  2   4      4     FAR-адрес ключа
;  6   8      4     FAR-адрес синхропосылки (S1,S2)
; 10  12      4     FAR-адрес таблицы замен
; 14  16      4     FAR-адрес области назначения
; 18  20      2     число шифруемых блоков
;-------------------------------------------------------
; При выходе: адрес области назначения и синхропосылка
; содержат скорректированные параметры, счетчик = 0
;-------------------------------------------------------
        .model  small C
        .code $gost
 extrn  gost:near
gammaLE proc
  public  gammaLE
; вычисляем смещение параметров в стеке
        a=8             ; 4 регистра в стеке
IF      (TYPE gammaLE) EQ OFFFEh
        a=a+2           ; если gamma - far-процедура
ENDIF
; Задаем  смещение  аргументов  в  стеке
;
key=    2+a
syn=    6+a
chg=    10+a
tar=    14+a
len=    18+a
;
; Сохранение регистров согласно соглашениям Turbo C
        push    BP
        push    DS          ; это нужно не всегда!
        push    SI
        push    DI
; Добавление  констант
        mov     BP,SP       ; настройка стекового кадра
        lds     SI,syn[BP]  ; DS:SI -> синхропопылка
        mov     AX,[SI]     ; AX <= младшее слово S1
        mov     DX,2[SI]    ; DX <= старшее слово S1
        mov     DI,4[SI]    ; DI <= младшее слово S2
        mov     CX,6[SI]    ; CX <= старшее слово S2
        lds     BX,chg[BP]  ; DS:BX <- адрес таблицы замен
; Готовим регистры и вызываем цикл 32-3
circle: les     SI,key[BP]  ; ES:SI <- адрес ключа
        mov     BP,CX       ; BP <- старшее  слово  S2
        mov     CX,32       ; CX <- число основных шагов
```

```
        call    gost        ; шаг  простой  замены
        mov     CX,BP       ; CX <- старшее  слово  S2
        mov     BP,SP       ; BP <- стековый  кадр
; Использование  сгенерированного  блока  гаммы
        les     SI,tar[BP]  ; DS:SI<-адрес  назначения
; будет альтернатива
        xor     DI,ES:[SI]      ; Гаммируем
        xor     CX,ES:2[SI]     ;    8-байтовый
        xor     AX,ES:4[SI]     ;       блок
        xor     DX,ES:6[SI]     ;          данных
        mov     ES:[SI],DI      ; и записываем
        mov     ES:2[SI],CX     ;   его
        mov     ES:4[SI],AX     ;     на нужное
        mov     ES:6[SI],DX     ;       место
        xchg    AX,DI           ; обмен значениями
        xchg    DX,CX           ;   S1 <--> S2
; альтернатива последним 10 строкам: короче, но медленнее
; на процессорах 8086,8088.
;       xor     ES:[SI],DI      ; Гаммируем
;       xor     ES:2[SI],CX     ;    8-байтовый
;       xor     ES:4[SI],AX     ;       блок
;       xor     ES:6[SI],DX     ;          данных
;       mov     AX,ES:[SI]      ; загружаем
;       mov     DX,ES:2[SI]     ;   блок для
;       mov     DI,ES:4[SI]     ;     следующего
;       mov     CX,ES:6[SI]     ;       шага
; альтернатива закончена
        add     word ptr tar[BP],8; коррекция указателя
        dec     word ptr len[BP]  ; коррекция счетчика
        jnz     circle      ; на генерацию нового блока
; заносим новое значение синхропосылки на его место
; (это нужно для обработки массива данных частями)
        lds     SI,syn[BP]  ; DS:SI -> синхропосылка
        mov     [SI],AX     ; заносим новые
        mov     2[SI],DX    ;    значения S1,S2
        mov     4[SI],DI    ;       на   их
        mov     6[SI],CX    ;          место !!!
; Восстановление регистров из стека
        pop     DI
        pop     SI
        pop     DS
        pop     BP
        ret
        gammaLE endp
        end                                              ■
```

Листинг 5

```
;-------------------------------------------------------
; Расшифровка массива данных в режиме гаммирования с
; обратной связью согласно алгоритму ГОСТ 28147-89
; Copyright (C) 1992-1994 Винокуров А.Ю.,  г. Москва.
;-------------------------------------------------------
; Параметры  при  вызове  передаются  в  стеке :
; Смещение Размер    П  А  Р  А  М  Е  Т  Р
;NEAR FAR
;  2   4      4     FAR-адрес ключа
;  6   8      4     FAR-адрес синхропосылки (S1,S2)
; 10  12      4     FAR-адрес таблицы замен
; 14  16      4     FAR-адрес области назначения
; 18  20      2     число шифруемых блоков
;-------------------------------------------------------
; При выходе: адрес области назначения и синхропосылка
; содержат скорректированные параметры, счетчик = 0
;-------------------------------------------------------
;
        .model  small C
        .code   $gost
 extrn  gost:near
gammaLD  proc
  public  gammaLD
; вычисляем смещение параметров в стеке
        a=8             ; 4 регистра в стеке
IF      (TYPE gammaLD) EQ 0FFFEh
        a=a+2           ; если gamma - far-процедура
ENDIF
; Задаем  смещение  аргументов  в  стеке
;
key=    2+a
syn=    6+a
chg=    10+a
tar=    14+a
len=    18+a
;
; Сохранение регистров согласно соглашениям Turbo C
        push    BP
        push    DS          ; это нужно не всегда!
        push    SI
  push    DI
; Добавление  констант
        mov     BP,SP       ; настройка стекового кадра
        lds     SI,syn[BP]  ; DS:SI -> синхропопылка
        mov     AX,[SI]     ; AX <= младшее слово S1
        mov     DX,2[SI]    ; DX <= старшее слово S1
        mov     DI,4[SI]    ; DI <= младшее слово S2
        mov     CX,6[SI]    ; CX <= старшее слово S2
        lds     BX,chg[BP]  ; DS:BX <- адрес таблицы замен
; Готовим регистры для вызова цикла 32-3;
circle: les     SI,key[BP]  ; ES:SI <- адрес ключа
        mov     BP,CX       ; BP <- старшее  слово  S2
        mov     CX,32       ; CX <- число основных шагов
        call    gost        ; шаг  простой  замены
        mov     CX,BP       ; CX <- старшее  слово  S2
        mov     BP,SP       ; BP <- стековый  кадр
; Использование  сгенерированного  блока  гаммы
        lds     SI,tar[BP]  ; DS:SI<-адрес  назначения
  xor     DI,[SI]     ; Гаммируем
        xor     CX,ES:2[SI] ;    8-байтовый
        xor     AX,ES:4[SI] ;       блок
        xor     DX,ES:6[SI] ;          данных
        xchg    ES:[SI],DI  ; меняем
        xchg    ES:2[SI],CX ;   его с
        xchg    ES:4[SI],AX ;     блоком
        xchg    ES:6[SI],DX ;       шифротекста
        xchg    AX,DI       ; обмен значениями
        xchg    DX,CX       ;   S1 <--> S2
        add     word ptr tar[BP],8; коррекция указателя
        dec     word ptr len[BP]  ; коррекция счетчика
        jnz     circle      ; на генерацию нового блока
; заносим новое значение синхропосылки на его место
; (это нужно для обработки массива данных частями)
        lds     SI,syn[BP]  ; DS:SI -> синхропосылка
        mov     [SI],AX     ; заносим новые
        mov     2[SI],DX    ;    значения S1,S2
        mov     4[SI],DI    ;       на   их
        mov     6[SI],CX    ;          место!!!
; Восстановление регистров из стека
        pop     DI
        pop     SI
        pop     DS
        pop    BP
        ret
gammaLD endp
  end                                                    ■
```

Листинг 6

```
;-------------------------------------------------------
; Выработка имитоприставки для массива данных согласно
; алгоритму ГОСТ 28147-89
; Copyright (C) 1992-1994 Винокуров А.Ю.,  г. Москва.
;-------------------------------------------------------
; Параметры  при  вызове  передаются  в  стеке :
; Смещение Размер    П  А  Р  А  М  Е  Т  Р
;NEAR FAR
;  2   4      4     FAR-адрес ключа
;  6   8      4     FAR-адрес исходной комбинации
; 10  12      4     FAR-адрес таблицы замен
```

```
   ; 14  16      4     FAR-адрес области назначения
   ; 18  20      2     число сворачиваемых блоков
   ;-------------------------------------------------------
; При выходе: адрес области данных и контрольная
; комбинация содержат скорректированные параметры,
; счетчик=0 ; DX:AX содержит младшее двойное слово
; контрольной комбинации, используемое обычно как
; имитоприставка, соответствует возврату типа Item
; (=unsigned long).
;-------------------------------------------------------- ;

        .model  small C
        .code $gost
  extrn  gost:near
imito   proc
        public  imito
; вычисляем смещение параметров в стеке
        a=8             ; 4 регистра в стеке
IF      (TYPE imito) EQ OFFFEh
        a=a+2           ; если gamma - far-процедура
ENDIF

; Задаем  смещение  аргументов  в  стеке
;
key=    2+a
syn=    6+a
chg=    10+a
tar=    14+a
len=    18+a
;

; Сохранение регистров согласно соглашениям Turbo C
        push    BP
        push    DS          ; это нужно не всегда!
        push    SI
        push    DI

; Загружаем значение комбинации от прошлого раза
        mov     BP,SP       ; настройка стекового кадра
        lds     SI,syn[BP]  ; DS:SI -> синхропопылка
        mov     AX,[SI]     ; AX <= младшее слово S1
        mov     DX,2[SI]    ; DX <= старшее слово S1
        mov     DI,4[SI]    ; DI <= младшее слово S2
        mov     CX,6[SI]    ; CX <= старшее слово S2
        lds     BX,chg[BP]  ; DS:BX <- адрес таблицы замен
; Добавляем блок данных
circle: les     SI,tar[BP]  ; DS:SI<-адрес  назначения
        xor     AX,ES:[SI]  ; Гаммируем
        xor     DX,ES:2[SI] ;    8-байтовый
        xor     DI,ES:4[SI] ;        блок
        xor     CX,ES:6[SI] ;           данных
; Готовим регистры и вызываем цикл 32-3
        les     SI,key[BP]  ; ES:SI <- адрес ключа
        mov     BP,CX       ; BP <- старшее  слово  S2
        mov     CX,16       ; CX <- число основных шагов
        call    gost        ; шаг  простой  замены
        mov     CX,BP       ; CX <- старшее  слово  S2
        mov     BP,SP       ; BP <- стековый  кадр
        add     word ptr tar[BP],8; коррекция указателя
        dec     word ptr len[BP]  ; коррекция счетчика
        jnz     circle      ; на генерацию нового блока
; заносим новое значение комбинации на его место
; (это нужно для обработки массива данных частями)
        lds     SI,syn[BP]  ; DS:SI -> синхропосылка
        mov     [SI],AX     ; заносим новые
        mov     2[SI],DX    ;    значения S1,S2
        mov     4[SI],DI    ;       на   их
        mov     6[SI],CX    ;          место!!!
; Восстановление регистров из стека
        pop     DI
        pop     SI
        pop     DS
        pop     BP
        ret
imito   endp
  end                                                    ■
```

Листинг 7

```
/*-------------------------------------------------------
  Построение расширенной таблицы замен (1024 байта) из
  таблицы замен (128 байтов) алгоритма  ГОСТ 28147-89.
  Copyright (C) 1992-1994 Винокуров А.Ю., г. Москва.
  -----------------------------------------------------
   Параметры  при  вызове:
     source - адрес таблицы замен в исходном формате
     target - адрес области для расширенной таблицы
  -----------------------------------------------------
*/
#pragma  inline
#include <dos.h>   // FP_SEG,FP_OFF

void    ExpCht (void far *source, void far *target)
{
// Настройка регистров для работы П/П
  asm   push    DS
  _DS=   FP_SEG  (source);
  _SI=  FP_OFF  (source);
  _ES=  FP_SEG  (target);
  _DI=  FP_OFF  (target);
  asm   {
   cld             // направление  просмотра
// Цикл  по  блокам  таблицы  замен
   mov     CX,4    // загрузить счетчик блоков
   }
blocks: asm     {
   mov     BX,SI   // BX<-адрес  начала  линии
   add     BX,10h  //     старших байтов блока
// Цикл  по  линиям  блока
   push    CX      // сохранить  счетчик блоков
   mov     CX,16   // загрузить  счетчик  линий
   }
lines:  asm   {
        push    SI      // сохранить  указатель  текущего
блока
   mov     AH,[BX] // AH <- старший    полубайт
   push    CX      // сохранить   счетчик линий
   mov     CL,4    // сдвиг полубайта  на место
   shl     AH,CL   //    старшего  полубайта
// Цикл  по  байтам  линии
   mov     CX,16   // загрузить счетчик  байтов
   }
bytes:  asm   {
   lodsb           // загрузить очередной  байт
   or      AL,AH   // добавить старший полубайт
   stosb           // ...  и записать результат
   loop    bytes   // цикл   по  байтам   линии
// Проверка  цикла  по  линиям
   pop     CX      // восстановить счетчик  линий
   pop     SI      // восстановить указатель
                   // текущего блока
   inc     BX      // продвинуть указатель байтов
   loop    lines   // цикл  по строкам  таблицы
// Проверка  цикла  по  блокам
   pop     CX      // восстановить счетчик блоков
   add     SI,20h  // продвинуть  указатель блока
   loop    blocks  // цикл по блокам
   }
   asm   pop     DS
  return ;
}                                                        ■
```

Листинг 8

```
/*-------------------------------------------------------
  Построение расширенного ключа зашифровывания (128 байтов)
  из ключа (32 байта) криптоалгоритма ГОСТ 28147-89.
  Copyright (C) 1992-1994 Винокуров А.Ю., г. Москва.
  -----------------------------------------------------
   Параметры  при  вызове:
```

```
     source - FAR-адрес ключа в исходном формате
     target - FAR-адрес области для расширения ключа
   Схема расширения 3+1: 3 раза вперед, 1 раз назад
  ------------------------------------------------------
*/
#pragma  inline
#include <dos.h>

void    ExpKey31 (void far *source, void far *target)
{
// Настройка регистров для работы П/П
  asm   push    DS
  _DS=   FP_SEG  (source);
  _SI=  FP_OFF  (source);
  _ES=  FP_SEG  (target);
  _DI=  FP_OFF  (target);
  asm   {
   cld              // направление  просмотра
; Три  прямых  блока
   mov     CX,16   // копировать
   rep movsw       //   первый
   sub     SI,32   //     блок
   mov     CX,16   // копировать
   rep movsw       //   второй
   sub     SI,32   //     блок
   mov     CX,16   // копировать
   rep movsw       //   третий блок
; один обратный блок
   mov     CX,8    // счетчик  двойных  слов
   }
xlp:    asm     {
   sub     SI,4    // указатель двойного слова
   mov     AX,[SI] // загрузить младшнее слово
   stosw           // ...  и  запомнить  его
   mov     AX,2[SI]// загрузить старшее слово
   stosw           // ...  и  запомнить  его
   loop    xlp     // цикл по двойным словам
   pop     DS
   }
  return ;
}
```

Листинг 9

```
/*-----------------------------------------------------
  gost.h - описания данных и прототипы функций шиф-
           рования
  -----------------------------------------------------
  Версия 1.00 от 25.06.94,  (c) 1994 Винокуров А.Ю.
  -----------------------------------------------------
*/

typedef unsigned long Item;  // элемент данных
typedef struct
{
  Item low,high;
} CrBlock;

void ExpCht (     // Расширение таблицы замен
  void far *source,      //   область исходной ТЗ
  void far *target);     //   область расширенной ТЗ

void ExpKey31 (         // Расширение ключа '3+1'
  void far *source,      //   область исходного ключа
  void far *target);     //   область расширенного ключа

void ExpKey33 (         // Расширение ключа '3+3'
  void far *source,      //   область исходного ключа
  void far *target);     //   область расширенного ключа

void ExpKey13 (         // Расширение ключа '1+3'
  void far *source,      //   область исходного ключа
  void far *target);     //   область расширенного ключа
```

```
void gamma  (           // Гаммирование данных
  void far *key,         //   адрес ключа
  void far *synchro,     //   адрес синхропосылки
  void far *xcht,  //   адрес таблицы замен
  void far *data,        //   адрес гаммируемых данных
  unsigned short blocks);//  число шифруемых блоков

void gammaLE(           // Зашифровывание-гаммирование с ОС
  void far *key,         //   адрес ключа
  void far *synchro,     //   адрес синхропосылки
  void far *xcht,  //   адрес таблицы замен
  void far *data,        //   адрес гаммируемых данных
  unsigned short blocks);//  число шифруемых блоков

void gammaLD(           // Расшифровка-гаммирование с ОС
  void far *key,         //   адрес ключа
  void far *synchro,     //   адрес синхропосылки
  void far *xcht,  //   адрес таблицы замен
  void far *data,        //   адрес гаммируемых данных
  unsigned short blocks);//  число шифруемых блоков

void simple(            // Шифрование простой заменой
  void far *key,         //   адрес ключа
  void far *data,        //   адрес шифруемых данных
  void far *xcht,  //   адрес таблицы замен
  unsigned short blocks);//  число шифруемых блоков

Item imito  (           // Вычисление имитоприставки
  void far *key,         //   адрес ключа
  void far *combine,     //   адрес начальной комбинации
  void far *xcht,  //   адрес таблицы замен
  void far *data,        //   адрес гаммируемых данных
  unsigned short blocks);//  число блоков
```

Листинг 10

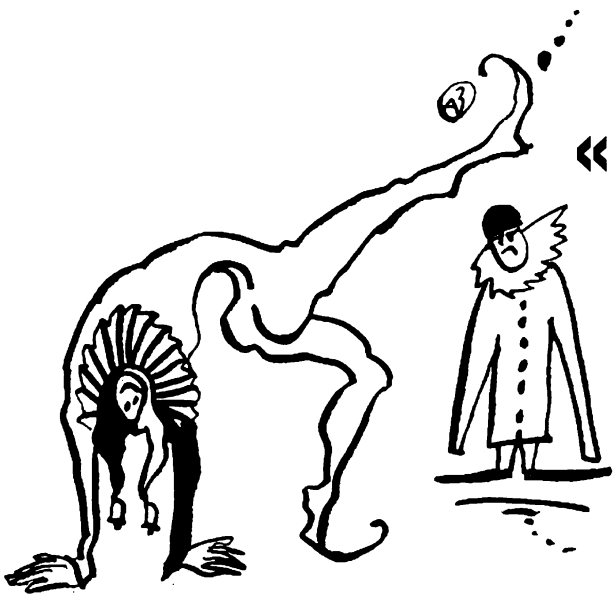
```
/*-----------------------------------------------------
  cryptor.c - исходный текст программы шифрования
              файлов
  -----------------------------------------------------
  Версия 1.00 от 25.06.94,  (c) 1994 Винокуров А.Ю.
  -----------------------------------------------------
  Эта программа специально составлена для публикации
  в журнале - см. описание применения.
  -----------------------------------------------------
*/

#include <io.h>          // open, filelength
#include <fcntl.h>       // O_RDONLY
#include <dir.h>         // MAXPATH
#include <ctype.h>       // toupper
#include <stdlib.h>      // strtoul
#include <stdio.h>       // printf
#include <memory.h>      // setmem
#include "gost.h"        // Данные и прототипы ГОСТа
// размер буфера обмена должен быть кратен 8
#define  BUFFER_SIZE    8192

// Глобальные статические данные

unsigned _stklen=BUFFER_SIZE+4096; // размер стека
typedef enum
{
  O_K_=0,           // все тип-топ
  errDoNotMatch,    // несовпадение имитоприставки
  errInvalidFlag,   // неверный флаг вызова программы
  errBadSynchro,    // неверно задана синхропосылка
  errBadImito,      // неверно задана имитоприставка
  errTooFiewParm,   // мало параметров вызова
  errTooManyParm,   // много параметров вызова
  errBadKey,        // плохой файл ключа
  errBadXCHT,       // плохой файл с таблицей замен
  errInvalidOper,   // неверная операция
  errSourceFailure, // ошибка в файле-источнике
```

```
    errTargetFailure, // ошибка в файле-приемнике
    errRenameFailure,   // ошибка при переименовании
    errDeleteFailure    // ошибка при удалении исходного файла
} errList;
unsigned char chtab[1024];
Item   Key[32];       // развернутый ключ
Item    Synchro[2]=   // синхропосылка
{
  0UL,0UL
};
Item   Imito[2]=
{
  0UL,0UL
};
Item    _Imito;
errList code=O_K_;      // код выхода
int   BackLink=  0;     // шифровать с ОС
int     ShowImito= 0;   // показать имитоприставку
int     CheckImito=0;   // проверить имитоприставку
int     Operation= 0;   // зашифровывание
int   ParmCount= 0;     // счетчик параметров
int     Deletion=  0;   // удаление исходного файла

char   *KeyFile=  NULL;// файл ключа
char   *ChtFile=  NULL;// файл таблицы замен
char   *InFile=   NULL;// исходный файл
char   *OutFile=  NULL;// файл результата

// Прототипы функций
errList setParam     (int argc, char **argv);
errList SignalError  (errList errType, char *errPlace);
errList loadCht      (void);
errList loadKey      (void);
errList   execute    (void);

// Главная программа модуля

errList   main(int argc, char *argv[])
{
  unsigned short i;
  printf("Программа шифрования данных с помощью "
    "криптографического алгоритма ГОСТ 28147-89\n"
    "Составил А.Ю. Винокуров, июнь 1994 года,"
    " свободное использование и копирование\n");

// Этот цикл строит тривиальную таблицу замен
  for (i=0; i<1024; i++) chtab[i]=i;
// Удалите его, если будете инициализировать  chtab

  return
      (code=setParam(argc,argv))!=O_K_ ? code :
   !InFile ? SignalError(errTooFiewParm,NULL) :
     (ChtFile && loadCht() != O_K_) ?
           SignalError(errBadXCHT,ChtFile) :
       (KeyFile && loadKey() != O_K_) ?
           SignalError(errBadKey,KeyFile) :
         execute();
}
// установка режимов согласно флагам вызова
errList   setParam(int argc, char **argv)
{
  for ( ; ++argv,--argc; )   // просмотр аргументов
  {
    char *c;
    if (**argv=='/')   // ключ командной строки
      switch (toupper(*(*argv+1)))  // выбор по букве
      {
      case 'D': Deletion=1;     break;
      case 'B': BackLink=1;     break;
      case 'N': BackLink=0;     break;
      case 'K': KeyFile=*argv+2;break;
      case 'C': ChtFile=*argv+2;break;
      case 'S':
   if (Synchro[0]=strtoul(*argv+2,&c,16),*c)
     return SignalError(errBadSynchro,*argv+2),1;
   break;
      case 'I':
   if (*(*argv+2))
     if (_Imito=strtoul(*argv+2,&c,16),*c)
       return SignalError(errBadImito,*argv+2);
     else CheckImito=ShowImito=1;
   else  ShowImito=1;
   break;
      default:
   return  SignalError(errInvalidFlag,*argv);
    }
   else switch(ParmCount++)    // другой аргумент
   {
     default:
   return  SignalError(errTooManyParm,*argv);
     case 0:         // декодируем функцию
   if (*(*argv+1))         // не одна буква
     return SignalError(errInvalidOper,*argv);
       switch(toupper(**argv)) // что за буква?
   {
     default:              // ошибка
       return SignalError(errInvalidOper,*argv);
         case 'D': Operation=1;// расшифровка
         case 'E':;            // зашифровывание
   }
   break;
     case 1:         // определяем источник
   InFile=*argv;   break;
     case 2:         // определяем приемник
   OutFile=*argv;
    }
  }
  return O_K_;
}

// Загрузка таблицы замен
errList loadCht(void)
{
  int   handle,FileSize;
  unsigned char  buffer[128];

  if ((handle=open(ChtFile,O_RDONLY))==-1 ||
       ((FileSize=filelength(handle)) != 128 &&
              FileSize != 1024) ||
    read(handle,
      FileSize==128 ? buffer : chtab,FileSize)==-1)
    return errBadKey;

  if (FileSize == 128) ExpCht(buffer,chtab);
  close(handle);
  return O_K_;
}
// Загрузка ключа
errList loadKey(void)
{
  int   handle,FileSize;
  unsigned char  buffer[32];

  if ((handle=open(KeyFile,O_RDONLY))==-1 ||
       ((FileSize=filelength(handle)) != 32 &&
              FileSize != 128) ||
     read(handle, FileSize==32 ?
           (void*) buffer : (void*) Key, FileSize)==-1)
    return errBadKey;

  if (FileSize==32) ExpKey31(buffer,Key);
  close(handle);
  return O_K_;
}
//  Осуществление шифрования

errList execute(void)
{
  int inHandle,outHandle,same;
  long FileSize;
  char buffer[BUFFER_SIZE];
// подготовка файлов
```

```
  simple(Key,Synchro,chtab,1);
  if (same=!OutFile)
    OutFile="$$$$$$$$.$$$";
  if((inHandle=open(InFile,O_RDONLY:O_BINARY))==-1 ::
     (FileSize=filelength(inHandle))== -1)
    return SignalError(errSourceFailure,InFile);
  if((outHandle=open(OutFile,
        O_WRONLY:O_CREAT:O_BINARY))==-1)
    return SignalError(errTargetFailure,OutFile);
  printf("%s шифрование файла %s\n",
    Operation ? "Рас" : "За", InFile);
// цикл шифрования
  for ( ; FileSize>0 ; FileSize-=BUFFER_SIZE)
  {
    unsigned short quant=min(BUFFER_SIZE,FileSize),
      blocks=(quant+sizeof(CrBlock)-1)/sizeof(CrBlock);

    if (read(inHandle,buffer, quant) == -1)
      return SignalError(errSourceFailure,InFile);
    if (!Operation && ShowImito)
    {
      if (FileSize % 8)
   setmem(buffer+FileSize,8-FileSize % 8,0);
      imito (Key,Imito,chtab,buffer,blocks);
    }
    (BackLink ? Operation ? gammaLD : gammaLE : gamma)
    (Key,Synchro,chtab,buffer,blocks);
    if (Operation && ShowImito)
    {
      if (FileSize % 8)
   setmem(buffer+FileSize,8-FileSize % 8,0);
      imito (Key,Imito,chtab,buffer,blocks);
    }
    if (write(outHandle,buffer, quant) == -1)
      return SignalError(errTargetFailure,OutFile);
  }
// завершающие действия
  close(inHandle);
  close(outHandle);
  if(ShowImito)
  {
    printf("Имитоприставка");
    if(CheckImito)
      printf(" %s совпадает !\n",
    Imito[1]==_Imito ? "" : " НЕ ");
    else
      printf("=%lx\n",Imito[1]);
  }
  if (Deletion :: same)
  {
    if (unlink(InFile)==-1)
      return SignalError(errDeleteFailure,InFile);
    if (same && rename(OutFile,InFile)==-1)
      return SignalError(errRenameFailure,InFile);
  }
  return O_K_;
}
// Выдача сообщения об ошибке
errList   SignalError(errList errType, char *errPlace)
{
  char *errText[]=
  {
  "Неверный флаг вызова программы - %s\n",
  "Неверно задана синхропосылка - %s\n",
  "Неверно задана имитоприставка - %s\n",
  "Необходим код функции и имя шифруемого файла\n",
  "Лишний параметр вызова - %s\n",
  "Неверно задан файл с ключом - %s\n",
  "Неверно задан файл с таблицей замен - %s\n",
  "Неверно задана выполняемая операция - %s "
              "(нужно E или D)\n",
  "Ошибка в исходном файле - %s\n",
  "Ошибка в файле результата - %s\n",
  "Невозможно переименовать файл $$$$$$$$.$$$ в %s\n",
  "Невозможно удалить файл %s\n"
  };
  printf(errText[errType-2],errPlace);
  return errType;
}
```

Листинг 11

```
.AUTODEPEND
# Определения транслятора, ассемблера, компоновщика
CC = bcc +CRYPTOR.CFG
TASM = TASM /MX /ZI /O
TLINK = tlink
# НЕявные правила
.c.obj:
  $(CC) -c {$<< }

.cpp.obj:
  $(CC) -c {$<< }
# Определенные символы
EXE_dependencies =  \
  cryptor.obj \
  expcht.obj \
  expkey31.obj \
  gost.obj \
  simple.obj \
  gamma.obj \
  gammald.obj \
  gammale.obj \
  imito.obj

# Явные правила
cryptor.exe: cryptor.cfg $(EXE_dependencies)
  $(TLINK) /v/x/c/LC:\BC\LIB @&&:
c0c.obj+
cryptor.obj+
expcht.obj+
expkey31.obj+
gost.obj+
simple.obj+
gamma.obj+
gammald.obj+
gammale.obj+
imito.obj
cryptor
      # no map file
emu.lib+
mathc.lib+
cc.lib
:
# Индивидуальные зависимости
cryptor.obj: cryptor.c
expcht.obj: expcht.c
expkey31.obj: expkey31.c
gost.obj: gost.asm
   $(TASM) GOST.ASM,GOST.OBJ
simple.obj: simple.asm
   $(TASM) SIMPLE.ASM,SIMPLE.OBJ
gamma.obj: gamma.asm
   $(TASM) GAMMA.ASM,GAMMA.OBJ
gammald.obj: gammald.asm
   $(TASM) GAMMALD.ASM,GAMMALD.OBJ
gammale.obj: gammale.asm
   $(TASM) GAMMALE.ASM,GAMMALE.OBJ
imito.obj: imito.asm
   $(TASM) IMITO.ASM,IMITO.OBJ
#      *Compiler Configuration File*
cryptor.cfg: cryptor.mak
  copy &&:
-mc
-C
-v
-y
-IC:\BC\INCLUDE
-LC:\BC\LIB
: cryptor.cfg
```



О пользе «ассемблерных извращений»

Д. Герасимчук

Прочитав статью «Ответ сердитому читателю» в «Мониторе» № 2.93, я бы хотел попытаться помочь редакции ответить на один из поставленных уважаемым Е. Гавришем вопросов относительно простой программы шифрования исполняемых файлов.

Я нашел довольно простое, как мне кажется, решение этой задачи. Программа шифрует COM-файлы с помощью операции XOR над данными файла и ключом, генерируемым случайным образом индивидуально для каждого файла. Решение это, как ни странно, стало возможным лишь при использовании «ассемблерных извращений».

В общем виде алгоритм работы программы таков.

1. Генерация ключа.
2. Запись в выходной файл подпрограммы автоматической расшифровки.
3. Последовательное кодирование и запись в выходной файл исходной программы.
4. Дозапись в выходной файл длины исходной программы.

Под исходной программой здесь понимается кодируемый файл в формате COM. Подпрограмма автоматической расшифровки — это небольшой ассемблерный фрагмент размером 89 байтов, который получает управление при запуске закодированной программы и, декодировав, передает ей управление (листинг 1). В листинге 2 приведен полный текст программы шифровки, написанной на Turbo Pascal 5.5.

Читатели могут самостоятельно поставить опыт по качеству шифрования. Для этого достаточно написать небольшую программу на ассемблере (что-то типа: mov ax,bx; mov bx,ax; ret), оттранслировать ее и затем попытаться взломать простую и защищенную копии программы при помощи Sourcer. Проделав это, вы, скорее всего, повторите слова известной рекламы: «Результат превосходит ожидания!». ■

Листинг 1

```
code segment
assume cs:code
org 100h
start:
; первая часть процедуры декодирования
  pushf ;сохраняем все, что только можно сохранить
  cli
  push ax
  push bx
  push cx
  push dx
  push ds
  push es
  push si
  push di
  mov dx,0c0dh ; сюда будет записана длина
               ; исходного файла
; вторая часть...
  mov si,0c0dh ; сюда запишем ключ
  mov cx,dx
  push cs
  pop es

again: dec cx ; декодируем...
  mov bx,cx
  add bx,0100h
  add bx,0c0dh ; здесь потом поставим
               ; длину подпрограммы
  dec bx ; расшифровки
  mov ax,es:[bx] ; AX=ES:[BX]
  xor ax,si ; дешифруем слово...
  mov es:[bx],ax ; и пишем его обратно
  loop again
  mov ax,0100h ; вычисляем адрес третьей
               ; части...
```

Листинг 1 (продолжение)

```
  add ax,0c0dh ; здесь потом поставим длину подпрограммы
  add ax,dx ; расшифровки
  push ax   ; запоминаем адрес перехода и...
  retn      ; прыгаем к третьей части
; третья часть дописывается в конец выходного файла
  push cs   ; DS=CS
  pop ds
  mov si,0100h ; готовимся к операции MOVSB
  add si,0c0dh ; и тут тоже поставим длину подпрограммы
  mov di,0100h ; дешифрования
  mov cx,dx
  cld
  rep movsb ; перемещаем декодированную программу по
  pop di ; адресу CS:100h, а затем восстанавливаем все,
  pop si ; что сохранили в первой части
  pop es
  pop ds
  pop dx
  pop cx
  pop bx
  pop ax
  popf
  mov ax,0100h ; готовимся к переходу CS:100h
  push ax
  mov ax,0000h ; чистим AX и BX и, наконец, ...
  mov bx,0000h
  retn ; передаем управление защищенной части
code ends
end start
```

■

Листинг 2

```
program Protect;
{$D-,R-,M 4096,16384,655360}
uses DOS,CRT;
{ в массивах хранятся соответствующие части подпрограммы
декодировки}
const
  PartOne:array[1..14] of Byte=
    ($9c,$fa,$50,$53,$51,$52,$1e,
    $06,$56,$57,$ba,$00,$00,$be);
  PartTwo:array[1..36].of Byte=
    ($8b,$ca,$0e,$07,$49,$8b,$d9,
    $81,$c3,$00,$01,$81,$c3,$34,
    $00,$4b,$26,$8b,$07,$33,$c6,
    $26,$89,$07,$e2,$ea,$b8,$00,
    $01,$05,$34,$00,$03,$c2,$50,
    $c3);
  PartThr:array[1..37] of Byte=
    ($0e,$1f,$be,$00,$01,$81,$c6,
    $34,$00,$bf,$00,$01,$8b,$ca,
    $fc,$f3,$a4,$5f,$5e,$07,$1f,
    $5a,$59,$5b,$58,$9d,$b8,$00,
    $01,$50,$b8,$00,$00,$bb,$00,
    $00,$c3);
procedure Fatal(Code:Byte);
begin
  Write('Фатальная ошибка:');
    case Code of
  1:WriteLn('пропущено имя файла.');
  2:WriteLn('расширение файла не .COM.');
  3:WriteLn('сбой операции ввода-вывода.')
    end;
  Halt(Code)
end;
procedure CheckIO;
begin
  if IOResult<>0 then Fatal(3)
end;
procedure BlockWriteWord(var F:File;Arg:Word);
{ процедура записи ассемблерного слова - сначала младший
байт, а потом старший }
var M:Byte;
begin
  M:=Lo(Arg);{$I-}BlockWrite(F,M,1);{$I+}CheckIO;
  M:=Hi(Arg);{$I-}BlockWrite(F,M,1);{$I+}CheckIO
end;
function ToUpper(S:String):String;
var I:Integer;
begin
  for I:=1 to Length(S) do
    S[I]:=UpCase(S[I]);
  ToUpper:=S
end;
const xFlag:Byte=0;
var FileName:String;
    tDir:DirStr;tName:NameStr;tExt:ExtStr;
    F1,F2:File;Passcode:Word;
    I,J:Word;Flag:Boolean;
    Get,Put:Word;
begin
  WriteLn('Программа защиты COM-файлов от ',
          'несанкционированного изменения.');
  WriteLn('Права на копирование принадлежат ',
          'Герасимчуку Д.');WriteLn;
  WriteLn('Введите имя файла для защиты либо ',
          'нажмите Ctrl-Break для останова.');
  Write(':');ReadLn(FileName);WriteLn;
  FSplit(FileName,tDir,tName,tExt);
  if tName='' then Fatal(1) else
  if (tExt<>'') and (ToUpper(tExt)<>'.COM') then Fatal(2)
else
  if tExt='' then tExt:='.COM';
  WriteLn('Введите имя выходного файла (расширение ',
          'COM не подразумевается).');
  Write(':');ReadLn(FileName);
  WriteLn;Assign(F1,tDir+tName+tExt);Assign(F2,FileName);
  {$I-}Reset(F1,1);{$I+}CheckIO;
  {$I-}Rewrite(F2,1);{$I+}CheckIO;
  Randomize;Passcode:=Random(65535);
  BlockWrite(F2,PartOne,SizeOf(PartOne),J);
  if J<>SizeOf(PartOne) then Fatal(3);
  BlockWriteWord(F2,Passcode);
  BlockWrite(F2,PartTwo,SizeOf(PartTwo),J);
  if J<>SizeOf(PartTwo) then Fatal(3);
  Flag:=True;I:=0;while Flag do begin
    BlockRead(F1,Get,2,J);if J<>0 then begin
      Inc(I);Inc(I);
      if J=1 then begin xFlag:=1;Dec(I) end;
      Put:=Get xor Passcode;
      BlockWrite(F2,Put,2,J);if J<>2 then Fatal(3);
      Write('Обработано байтов:',I);
      GoToXY(1,WhereY)
    end else Flag:=False
  end;WriteLn;WriteLn;if xFlag=1 then Inc(I);
  BlockWrite(F2,PartThr,SizeOf(PartThr),J);
  if J<>SizeOf(PartThr) then Fatal(3);
  {$I-}Seek(F2,11);{$I+}CheckIO;
  BlockWriteWord(F2,I);
  Close(F1);Close(F2);
  WriteLn('Файл успешно зашифрован.')
end.
```

■

Что такое «модем для России»?

Д. Мирошников, С. Воеводский, Д. Воеводский

Попробуем остановиться на критериях, которые отличают модем, изготовленный специально для России, от стандартного.

Адаптация модема к телефонной сети СНГ — нужна ли она?

Вопрос адаптации модемов стал ныне весьма популярным, причем подход к нему со стороны участников рынка модемов различный.

Некоторые дистрибьюторы западных фирм вообще отрицают наличие проблемы: «У нас фирменные модемы, они ни в какой адаптации не нуждаются — и так хорошие». В качестве аргумента обычно приводятся исследования западных компьютерных журналов. Что же, безусловно, успех модема на Западе говорит о его хороших потребительских качествах. Однако, если посмотреть методику тестирования модемов западными журналами, то можно заметить, что основное внимание уделяется таким параметрам, как скорость передачи файлов, удобство программирования, эргономика, экономичность, но не проблеме **установления соединения** и **поддержания связи** как таковой. И это не удивительно, так как при высоком качестве телефонных каналов проблем с установлением соединения не бывает.

В России же основное требование к модему совсем иное. При низком качестве телефонных линий главным для потребителя является то, чтобы его модем как можно лучше **устанавливал соединение** и уверенно **поддерживал связь**. И это понятно: какой прок от самого быстрого модема, если он «не слышит несущую» или то и дело разрывает связь из-за импульсных помех. Довольно неприятной для потребителя является и нечувствительность многих западных модемов к станционным зуммерам «готовность», «занято», «звонок». Если «занято» в конечном итоге и не является камнем преткновения, модем лишь чуть-чуть дольше дозванивается до абонента, то неуверенное распознавание звонка делает невозможным использование модема в режиме автоответа (HOST), а нечувствительность к «dialtone» делает невозможным режим автоматической рассылки сообщений.

Таким образом, отрицание самой необходимости адаптации модемов к сети СНГ является скорее всего лишь следствием невозможности для того или иного поставщика должным образом усовершенствовать свою продукцию.

Если нам удалось вас убедить, что адаптация необходима, попробуем рассказать, в чем же она **возможна**. Отбросим сразу вопросы применения различных помехозащищенных протоколов, таких как «модифицированный» V.23, PEP и другие. Безусловная польза этих протоколов в режиме «точка-точка» не вызывает сомнений, вспомним недавний успех «вездеходных» модемов LEXAND, АНАЛИТИК, UNICOM, МАСТАК, поддерживавших модифицированный V.23. Однако тенденции развития рынка показывают, что сегодня в России, как и во всем мире, модем работает в сети, а значит, на «другом конце» находится стандартный модем, поддерживающий международно признанный и распространенный протокол.

Что же тогда можно улучшать в модеме? Как увеличить его устойчивость и удобство в работе? Попробуем описать некоторые методы, применяемые нами на практике.

Статья публикуется на коммерческой основе.

Основные пути адаптации модемов

Адаптацию можно понимать по-разному. Попробуем описать различные способы улучшения потребительских качеств модемов по принципу «от простого к сложному». Как примеры реализации того или иного метода мы будем рассматривать модемы, поставляемые нашей фирмой (НТЦ «НАТЕКС») и другими российскими компаниями.

1. Адаптация программного обеспечения (software), поставляемого с модемом, хорошая техническая поддержка и документация.

Это, безусловно, самый простой и необходимый метод адаптации. Действительно, какой пользователю толк от факсовой программы, если она вместо русских символов передает «арабскую вязь». Русифицированная факсовая математика широко распространена (BITFAX, русские шрифты к MTEZ) и обязательна должна предлагаться пользователю при покупке модема. Кроме того, в модеме должны быть либо записаны при изготовлении, либо хотя бы приложены на бумаге начальные установки, пригодные для России. Неквалифицированный пользователь (а абонентский модем — устройство не для профессионалов), конечно, не должен ломать голову, почему модем не набирает номер, если к его (пользователя) несчастью в нем задан заводской профиль с тональным набором.

Почти все российские компании, профессионально занимающиеся поставками модемов, осуществляют адаптацию или локализацию устройств такого типа. В качестве положительных примеров русскоязычной модемной и факсовой математики можно привести пакет XFREE разработки НПП «ФАКТОР» и систему FAXLINE/WINDOWS фирмы «STEEPLER».

2. Адаптация микропрограмм, записанных в ПЗУ модема (firmware).

Микропрограмма сигнального процессора — основа функционирования модема. Протоколы канального уровня, процедура отработки команд, сжатие данных и коррекция ошибок — все это в значительной степени определяется микропрограммой.

Что можно и нужно адаптировать в микропрограмме модема?

Во-первых, подпрограммы анализа станционных сигналов (зуммеров). Модем должен надежно распознавать сигнал «готовность» (DIALTONE), так как в противном случае невозможен автоматический набор, например, при междугородных соединениях. Сигнал «занято» (BUSY) менее важен для практической работы, хотя, конечно же, уверенное его распознавание делает работу с модемом более приятной. Наконец, 100% распознавание звонка (RING) абсолютно необходимо для узловых или host-модемов.

Во-вторых, микропрограмма должна обеспечивать выполнение некоторых специфических команд, полезных в условиях России, как-то: возможность регулировки выходного уровня, ручной настройки на протоколы, диагностика линии и другие.

Корректирование firmware — важный и трудный шаг. И, конечно, очень немногие фирмы способны его сделать. Подавляющее большинство импортных модемов продается в России со стандартной микропрограммой и, как следствие этого, в 90% случаев неуверенно распознает «занято» и не имеет регулировки уровня, а иногда (например, в случае применения в модеме микросхем AT&T 14400 бит/с) не определяет также и «готовность» (DIALTONE).

В качестве примеров адаптации микропрограмм можно привести изделия фирмы INPRO. Специалисты компании ввели в модем даже такую «специфически русскую» функцию как АОН. Среди модемов серии UNICOM, поставляемых нашей фирмой, многие модели также имеют модифицированную микропрограмму. Например, в модеме UNICOM 9624VQE с помощью модификации микропрограммы удалось добиться практически 100% распознавания зуммера «занято», обеспечить возможность повышения выходного уровня, ввести команду индикации уровня принимаемого сигнала.

3. Адаптация аппаратной части модема является самой сложной и, естественно, наиболее существенно влияет на качество работы модема.

Аппаратная адаптация должна включать в себя, во-первых, выбор компонентов (набор БИС), наибольшим образом приспособленных к эксплуатации в сетях СНГ. Не секрет, что параметры модема на 90% определяются его основой — сигнальным процессором. Поэтому выбор комплекта — «главный метод адаптации». Комплект, прекрасно зарекомендовавший себя на Западе, может быть малопригодным для российских условий.

Во-вторых, в модеме должны быть предусмотрены аппаратные методы повышения чувствительности (специальная схема сопряжения с линией, минимизация собственных шумов модема). При большом затухании в линии стандартных -43 dB чувствительности может не хватить и модем «не услышит несущую», а соответственно и не войдет в связь.

В-третьих, модем должен работать с широким диапазоном входных сигналов — иметь широкий динамический диапазон. Кроме того, чрезвычайно удобным является наличие в модеме переключателей, позволяющих конфигурировать модем без помощи AT команд. Такая функция становится крайне необходимой при использовании модема с аппаратурой X.25 отечественного производства. В этом случае модем должен входить в соединение сразу по включению питания, без вмешательства оператора и не дожидаясь каких-либо команд от ТК или ЦКП. Как ни странно, такой режим — очень большая редкость среди импортных модемов, поэтому наличие данной функции (для простоты будем называть ее DUMB-режим) также можно причислить к адаптации, имея в виду адаптацию к широко распространенной сетевой аппаратуре.

У большинства российских дистрибьюторов выбор комплекта микросхем определяется фирмой-поставщиком. Поэтому часты случаи, когда прекрасно работавший в России модем снимается западным производителем с производства и заменяется новой версией — более технологичной, экономичной и дешевой, но... не работающей в России. Какой же выход? Мы предлагаем следующий: если нельзя ориентироваться на модели одной фирмы, то почему бы не работать с разными поставщиками? Например, мы предлагаем клиенту три типа модемов, имеющих, казалось бы, совершенно одинаковые характеристики — скорость 14400 бит/с, аппаратную коррекцию ошибок и сжатие данных. Зачем? Для того чтобы иметь возможность поставки оптимального оборудования для каждой из трех коммуникационных задач: абонентский модем, host-модем и магистральный. Остановимся более подробно на особенностях каждого типа оборудования. Это модемы UNICOM 1414VQE, BULLET E1414MX и TAICOM 9642+.

Модем UNICOM 1414VQE основан на микросхемах Rockwell, которые обладают хорошей чувствительностью, устойчивы на линии, недороги, легко адаптируются для распознавания станционных зуммеров. Однако микросхемы Rockwell имеют и серьезные недостатки: ограниченный динамический диапазон, что не позволяет поднять уровень выходного сигнала, строго заданный набор функций host-процессора, не поддающийся расширению, склонность к «зависанию» при бросках питания и мощных импульсных помехах в линии. Все эти недостатки существенны для модемов, используемых как host или в качестве магистральных, но совершенно незаметны при использовании модема в качестве абонентского, то есть вызывающего узел сети. Следовательно, UNICOM 1414VQE мы предлагаем абонентам сетей. Адаптация этой модели, естественно, также направлена на повышение удобства работы абонента: 100% распознавание «занято», высокая чувствительность (-45...-49 dB), простота в работе и привлекательный внешний вид. Естественно, важным критерием в этом классе является и цена, которая для UNICOM 1414VQE минимальна.

Модем Bullet E1414MX основан на AT&T с адаптированной микропрограммой. Этот модем дороже UNICOM 1414VQE и поэтому менее эффективен с точки зрения стоимости. Зато для узлов сетей Bullet на порядок удобней. Благодаря использованию в качестве host-процессора микропроцессора ZILOG в Bullet предусмотрено множество дополнительных функций: удаленное конфигурирование, защита от несанкционированного доступа, расширенная диагностика линии. Вместе с тем сигнальный процессор AT&T наряду с высокой чувствительностью (-45 dB) обеспечивает широкий динамический диапазон, что делает возможным увеличение выходного уровня до -4 dB. Таким образом, Bullet оптимален как узловой модем, а модификация E1414MX DUMB прекрасно зарекомендовала себя и на магистральных каналах.

Наконец, **TAICOM 9642+** — самый дорогой из трех модемов, рассчитанный на применение в качестве магистрального и узлового в сетях передачи данных. TAICOM основан на специализированном процессоре Rockwell, существенно более дорогом, чем «ширпотребовский» вариант, используемый, например, в UNICOM. Что же делает этот модем столь привлекательным для профессиональных пользователей? Во-первых, чрезвычайно широкий динамический диапазон: уровень выходного сигнала может быть увеличен до 0 dB, что очень полезно на магистральных каналах. Во-вторых, полное конфигурирование модема с помощью DIP-переключателей, что делает возможным его использование с любыми типами ЦКП и ТК и исключает сбои в работе сети из-за вмешательства неквалифицированного пользователя. В-третьих, TAICOM способен работать не только по двух-, но и по четырехпроводным выделенным каналам. Наконец, «крепкий» металлический корпус и вариант конструкции RACK MOUNT определяют надежность и удобство использования модема в сетях ПД. Строго говоря, функции TAICOM выполняют и многие из универсальных дорогих модемов, но разумно ли платить в 3 раза дороже за универсальность, которая в 90% случаев не используется?

Рассмотрев проблему адаптации модемов к сети СНГ на примерах модемов, поставляемых фирмой «НАТЕКС», мы, конечно, не умаляем достоинств модемов других поставщиков. Единственное, что мы хотели показать, — это необходимость реальной адаптации модемов к сети СНГ и возможность для российских поставщиков осуществить ее,не теряя конкурентоспособности. ■

Подписку на журнал «Монитор»
на 1995 год
можно оформить
в любом отделении связи по каталогам
агентства «Книга-Сервис»
(индекс журнала «Монитор» 099)
и фирмы «Информ»,

в фирме «Информ»: 113447 Москва, а/я 3,
тел. (095) 129-68-29,

в агентстве «Книга-Сервис»: 117168 Москва,
ул. Кржижановского, д. 14, корп. 1,
тел. (095) 129-29-09, 124-94-49, 129-72-12.

Для оформления подписки через агентство «Книга-Сервис» организации перечисляют стоимость подписки на расчетный счет агентства и высылают в адрес агентства копию платежного поручения и заявку с указанием издания, количества комплектов, полного почтового адреса и фамилии, имени, отчества подписчика. Частные лица отправляют деньги почтовым переводом, указав в графе «для письменного сообщения» названия изданий и количество комплектов. Издания будут доставляться по указанному подписчиком обратному адресу.

Банковские реквизиты агентства «Книга-Сервис»:

р/с 7467374 в РосстромБанке,
к/с 161324 в РКЦ ГУ ЦБ РФ по г.Москве,
МФО 44583001, уч. 83.

Для тех, кто имеет возможность забирать журналы в редакции, можно оформить подписку в АОЗТ «Монитор». Для этого надо приехать к нам, заполнить купон, заплатить по наличному расчету. Цена годовой подписки 10$ в рублях по курсу ММВБ на день оплаты.

Адрес редакции:

Москва, ул. Большая Якиманка, 38А, 2 этаж.
Телефон: 238-99-11.
Факс: 238-99-11, E-mail: post@mntr.msk.su.

Михаил Сычев принимает посетителей по вопросам подписки по четвергам с 11 до 17 часов.

Редакция журнала «Монитор» предлагает следующие выпуски «Монитор-Диск»:

	Цена, $
Выпуск 1. Полные тексты статей и листингов номеров за 1991-1992 годы (10 номеров)	2.00 за комплект
Выпуск 2. Аннотации статей и полные тексты листингов номеров за 1993 год (8 номеров)	2.00 за комплект
Выпуск 3. Аннотации статей и полные тексты листингов номеров за 1994 год (8 номеров)	3.20 за комплект
Выпуск 4. Аннотации статей и полные тексты листингов номеров за 1995 год	0.40 за один номер

Цены приведены в долларах США без учета стоимости магнитных носителей. Оплата в рублях по курсу ММВБ. Редакция не имеет возможности форматировать дискеты и делать выборку информации.

Обращаться в редакцию:
Москва, ул. Большая Якиманка, д. 38 А (2 этаж), тел: 238-99-11.

НАДЕЖНАЯ СВЯЗЬ
НЕ ВСЕГДА ДОРОГО

Модемы

Полный ряд модемов и факс-модемов от простейших до профессиональных и со скоростью до 28800 бит/с. Схемотехника и микропрограммы, разработанные ***НАТЕКС*** *совместно с американской фирмой-производителем, обеспечивают надежную связь на телефонных каналах СНГ.*

Пользовательские модемы ***UNICOM****, полностью адаптированные для отечественных сетей.*

Профессиональные модемы ***BULLET*** *с расширенным набором функций.*

Магистральные 2-х и 4-х проводные модемы ***TAICOM****, скорость от 14400 до 128000 бит/с.*

Поддержка систем голосовой почты и аппаратуры X.25 и X.400. Сертификат Министерства Связи России и ведущих телекоммуникационных сетей.

Терминалы

VERTEX, NVT - недорогие видеотерминалы для систем UNIX, IMS MDOS7, DOS Line, PC-MOS, PICK, VM/386, MUMPS, MSM ...

- *Эмуляция - VT52, VT100, VT220, PC TERM*
- *Монитор - Hercules, Mono VGA, Color VGA*
- *Кодировка - альтернативная, КОИ-8, ISO-8859*
- *Скорость обмена от 110 до 115200 бод.*
- *Интерфейсы - RS232C и Centronics*

Мультиплексоры

AST4, PCCOM8 *-* **RS232C** *наиболее популярные и недорогие 4-х и 8-ми канальные мультиплексоры. Позволяет устанавливать любой стандартный базовый адрес, переключатель инверсный/неинверсный регистр прерывания.*

HICOM 4, 8 *- скоростные варианты AST4 и PCCOM8 с внутренним буфером (16C550UART).*

ТЕЛЕКОММУНИКАЦИОННЫЕ СЕТИ

Постака, разработка
и производство
под "ключ"

НТЦ "НАТЕКС"

тел. (095) 325-0088, 325-1862;
факс (095) 325-2293

Дилерам большие скидки

МОНИТОР 1.95

Аспект

Технологии

Звуки мультимедиа в среде MS-DOS

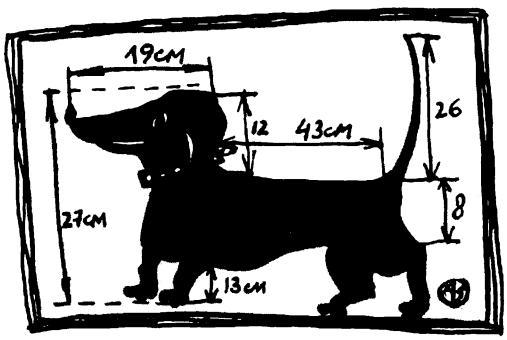
Аппаратура

Методы оценки производительности

Программы

Удаленный доступ: The Norton pcANYWHERE

Методы оценки производительности вычислительных систем



Д. Волков

Появление множества разнообразных архитектур компьютеров класса рабочих станций и серверов требует новых способов их оценки, отражающих возможности не только процессора, но и всей архитектуры в целом.

Согласно «скромным» рекламным заявлениям отдельных фирм — производителей компьютерных систем, каждое их чадо непременно является вундеркиндом и способно справиться с любой задачей. Для проверки подобных утверждений используются специальные тестовые наборы, или benchmarks, позволяющие упростить болезненный процесс выбора конфигурации, отвечающей конкретным требованиям. Сегодня почти полностью отсутствует информация по данному вопросу. Многие публикации в компьютерной прессе опираются на данные, полученные от заинтересованных фирм, или отражают мнение авторов, которое часто не соответствует объективной оценке возможностей конкретной компьютерной системы. Однако последствия скоропалительного выбора могут выходить далеко за рамки просто потраченных средств. По данным корпорации 3M, за год работы каждого компьютера одна неделя расходуется на восстановление данных, потерянных в результате ненадежной работы аппаратуры. Согласно разным оценкам, это составляет от 10 до 100 тыс. долл. По мнению фирмы IDC на восстановление двадцати мегабайтов данных в инженерных подразделениях расходуется до 98 тыс. долл. и 42 рабочих дня. Эти цифры свидетельствуют, что стоимость потерь часто может намного превышать стоимость самой системы, поэтому полезно всегда стремиться к покупке компьютеров, стоимость которых адекватна возможным потерям, что, впрочем, уже хорошо себе уяснили многие отечественные банковские учреждения.

Сегодня наиболее распространены оценки MIPS (миллион инструкций в секунду при работе с целыми числами) и MFLOPS (миллион инструкций в секунду при работе с вещественными числами). Однако с появлением RISC-архитектур, а также новых системных решений, продемонстрировавших, что узким местом может быть совсем не производительность процессора, а, например, скорость шины или быстродействие операций ввода/вывода, старые методики оценки перестали отражать реальные возможности архитектуры в целом. Каждая конкретная задача предъявляет к вычислительной системе свои требования по обработке больших массивов данных, производительности вычислений с вещественными числами или по организации работы множества пользователей.

Оценка производительности процессора

Тестовый набор **Dhrystone**, созданный в 1984 г. Рейнхольдом Вейкером (Reinhold Weicker), предназначался для оценки производительности процессора при выполнении операций с целыми числами. Тест состоит из 100 команд: 53 — операции присваивания, 32 — управления, 15 — вызовы функций. Результат измерения по данному тесту получается при делении количества инструкций, выполняемых в секунду, на число 1757. Один MIPS равен 1757 Dhrystone в секунду, что эквивалентно производительности компьютера VAX 11/780.

Тестовый набор **Linpack** (MFLOPS) предназначен для измерения производительности компьютера при решении системы линейных уравнений. Обычно применяется четыре набора тестов: решение с одинарной и двойной точностью системы уравнений размером 100 × 100 и системы 1000 × 1000. Тест позволяет получить оценку эффективности процессора и компиляторов при обработке вещественных чисел. Основу программы на языке Фортран составляют функции из библиотеки Basic Linear Algebra Subroutines (BLAS). Результат измеряется в MFLOPS — миллион операций, выполняемых над вещественными чис-

лами в секунду. Деятельность по развитию теста Linpack, проведению тестирования и распространению отчетов финансируется независимой правительственной организацией — министерством энергетики США. Последнее обстоятельство выгодно отличает данные Linpack от оценок, публикуемых фирмами-производителями. Приведем таблицу сравнения некоторых вычислительных систем по тестам Dhrystone и Linpack (табл. 1).

Высокий потенциал при работе с вещественной арифметикой процессоров PA-RISC, используемых в рабочих станциях фирмы «Hewlett-Packard», объясняет причину широкого применения моделей 735 и 755 в научно-исследовательских целях. Процессоры R4400, R4600 и R8000 фирмы MIPS имеют высокие показатели быстродействия при работе с целочисленной арифметикой, что является причиной успеха серверов серии 4000 фирмы «Control Data Systems» и компьютеров класса Challenge фирмы «Silicon Graphics».

В данную таблицу не случайно попал компьютер Exemplar фирмы «Convex», известной исключительно как производитель суперкомпьютеров. Последние архитектурные решения, предложенные «Convex» совместно с фирмой HP, позволяют получить машину с производительностью суперкомпьютера и ценой старших моделей рабочих станций (от 100 тыс. долл.). Использование в качестве «строительного кирпича» процессоров PA-RISC обеспечивает теперь полную двоичную совместимость всех моделей, начиная от клона станций HP до мощных действительно суперкомпьютеров Convex, состоящих из 128 процессоров, имеющих память 31 Гб и производительность 26 Гфлоп.

По тесту Linpack лидерами, на текущий момент, являются процессоры POWER PC, PA-7100 и Alpha. Состояние на осень 1994 года отражает выборочная таблица результатов тестирования по оценкам Linpack (табл. 2).

Из-за постоянных изменений во взаимоотношениях различных производителей компьютерной техники: HP и «Convex», «Motorola» и PowerPC и так далее, первенство в данной таблице оказывается очень неустойчивым. Следует учесть, что Linpack характеризует пиковую производительность процессора и для оценки особенностей межпроцессорных взаимодействий требуется дополнительный анализ. Очевидно также, что простое умножение производительности одного процессора на их число не дает реальную производительность многопроцессорной системы.

Список компьютеров фирмы HP возглавляет модель 735, открывающая в зависимости от конфигурации ценовую категорию компьютеров от 50 до 100 тыс. долл. Далее в диапазоне 39 — 26 MFLOPS мелькают, чередуясь, различные модели DEC, IBM PowerPC, которые плавно переходят в разряд рабочих станций средней производительности, где выделяются компьютеры фирмы HP.

Отметка 10 MFLOPS примечательна появлением систем на базе процессора Intel860, за которыми до значения 2,4 MFLOPS размещаются известные в прошлом суперминикомпьютеры ApolloDN10000, VAXstation 4000 и другие.

Первые представители старших моделей любимого российскими пользователями клона персональных компьютеров следуют в таблице после показателя 2,4 MFLOPS. Закрывает таблицу IBM PC/AT/286/287.

Сравнение по оценкам Dhrystone и Linpack является только предварительной прикидкой, подготавливающей почву для ответа на основной вопрос — как каждая из перечисленных компьютерных систем поведет себя при решении реальных задач? При этом важной может оказаться не только способность компьютера проводить вычисления, но и манипулировать большими массивами данных, одновременно

Таблица 1

Модель	Dhrystone, MIPS	Linpack 100, MFLOPS	Linpack 1000, MFLOPS
Convex Exemplar SPP1000	-	800	-
HP 735 (PA-7150/125)	154	57	120
CD 9140 (R4400/150)	120	26	75
Sun 10/41 (SuperSPARC/40)	109,5	7	23

Таблица 2

Модель	Linpack 100, MFLOPS
CRAY C90 (16 процессоров)	479
NEC SX-3/14R (1 процессор)	368
IBM Power2 990 (71,5 МГц)	140
CONVEX C-4/XA-2 (2 процессора)	130
SGI Power CHALLENGE (1 процессор)	101
DEC 7/650 Alpha (5 процессоров)	44
HP 9000/735 (99 МГц)	41
Sun SPARC 10/52 (1 процессор)	23
Intel iPSC/860 (1 процессор)	9,8
Gateway 2000 66Мгц(80486DX2)	2,4
Apple Macintosh Quadra 950	2
Compaq Deskpro 80486/33 w/487	1,4
IBM PC 286/287	0,0091
IBM PC 8087	0,0069

управлять работой нескольких задач или реагировать на воздействия от других компьютеров.

Новым шагом на пути создания более информативных оценок возможностей процессора явился тестовый набор корпорации SPEC (Standard Performance Evaluation Corp.). Учредителями SPEC выступили 19 фирм, среди которых были «Hewlett-Packard», «Control Data Systems», MIPS. Основным результатом деятельности корпорации являются наборы SPECint92 и SPECfp92, оценивающие работу процессора с целыми и вещественными числами соответственно.

Производительность процессора по **SPECint92** измеряется в относительных единицах к компьютеру VAX 11/780 и равна среднеарифметическому по следующим тестам.

Espresso — генерация и оптимизация матриц булевой логики. Эвристики по минимизации логических выражений.

Li — программа на языке Си, реализующая интерпретатор с языка Лисп. Используется для оценки процессора в режиме интенсивной работы.

Eqntott — программа на языке Си, преобразующая логические уравнения в таблицы истинности. До 95% всего времени работы данного теста используется библиотечной программой сортировки qsort().

Compress — стандартная утилита UNIX по двадцатикратной упаковке текстового файла размером 1 Мб на основе алгоритма Lempel-Ziv. Данная программа применяется для оценки работы процессора в интенсивном режиме с одновременным выполнением операций ввода/вывода.

Sc — программа на Си, реализующая операции по обработке электронных таблиц: перемещение курсора и данных, управление файлами, операции со строками и столбцами и тому подобное.

Gcc — стандартный компилятор GNU языка Си, преобразующий 19 исходных текстов программ в оптимизированный ассемблерный код.

Набор **SPECfp92** состоит из следующих тестовых программ.

Spice2g6 — программа на Фортране для моделирования аналоговых цепей. Тест выполняется с двойной точностью, интенсивно использует кэш-память и работает с комплексными числами. До 80% всех операций занимают пересылки из памяти в память.

Doduc — программа на языке Фортран для моделирования методом Монте-Карло термодинамики ядерного реактора. Тест выполняется с двойной точностью, содержит множество коротких ветвлений и циклов.

Mdljdp2 — программа на Фортране по решению с двойной точностью уравнения движения для 500 атомов в квантовой химии. С помощью уравнений статистической механики вычисляется энергия и давление газа.

Wave5 — программа на Фортране для решения уравнения Максвелла и управления движением частицы в Декартовой сетке при различных граничных условиях. Тестовый пример содержит 500 тыс. частиц на сетке из 50 тыс. точек, расположенных с шагом 5 единиц.

Tomcatv — программа на Фортране для нанесения сетки на модель корпуса автомобиля или крыла самолета. Тест выполняется с двойной точностью и специально ориентирован на проверку возможностей использования кэш-памяти и организации вычислений для суперскалярных и векторных компьютерных систем.

Ora — программа на языке Фортран по трассировке лучей, проходящих через оптическую систему из сферических и плоских поверхностей. Проверяется работа процессора в интенсивном режиме обработки чисел в формате с двойной точностью.

Alvinn — программа на Си по моделированию движения роботов. На входе программа получает сетку из 1220 узлов, в каждом из которых помещаются данные от видеокамеры и дальномера. Программа генерирует 35 различных управляющих директив самоходной тележке робота.

Ear — программа на Си по моделированию уха человека. Тест выполняется с двойной точностью. На входе программа получает файл с данными, задающими звуковое окружение человека, а на выходе формирует спектр разложения звука по шкале времени.

Mdljsp2 — версия программы Mdljdp2 для одинарной точности.

Swm256 — программа на Фортране по решению уравнения shallow-water для сетки 256 × 256. Тест выполняется с одинарной точностью.

Su2cor — векторизированная программа на Фортране по решению с двойной точностью задачи квантовой физики. Вычисление массы элементарных частиц по теории Quark-Gluon на основе метода Монте-Карло и данных из статистической механики.

Hudro2d — векторизированная программа на Фортране по нахождению параметров межгалактического газа с помощью уравнений Навье-Стокса. Тест выполняется с двойной точностью.

Nasa7 — набор из семи библиотечных модулей на языке Фортран, выполняемых с двойной точностью (умножение матриц, инвертирование матриц и так далее).

Fpppp — программа на Фортране по моделированию процесса замещения атомов на сериях Гаусса в квантовой химии. Задача трудно распараллеливается из-за наличия больших блоков данных. Тест выполняется с вещественными числами двойной точности в условиях интенсивного ввода/вывода.

Анализируя состав задач, включенных в смеси SPEC, можно сделать предварительные выводы о принципиальной пригодности процессора для решения уже конкретного круга задач (табл. 3).

Таблица 3 подтверждает вывод о предпочтительности компьютеров на базе PA-RISC для выполнения физических расчетов. Для решения задач управления обработки информационных потоков или создания среды разработки программ

можно использовать компьютеры на RISC-процессорах R4000, R4400, R8000. Наиболее полно возможности данных процессоров реализуются в серверах серии 4000 фирмы «Control Data» и Challenge фирмы «Silicon Graphics». Можно привести множество примеров использования станций CD4430, 4440, 4460 InfoServer как в качестве машин для рабочих мест программиста, так и в качестве серверов баз данных в России и за рубежом.

Количество и качество

Для определения пропускной способности компьютерной системы используется оценка SPEC Homogeneous Capacity Method, или SPECrate. В отличие от предыдущих тестов, измеряющих скорость выполнения одного задания, **SPECrate** позволяет определить, сколько заданий способен выполнить компьютер за единицу времени.

Если отдельные однопроцессорные системы выполняют одну задачу быстрее многопроцессорных, то многопроцессорные комплексы могут обрабатывать больше заданий в единицу времени. Если один процессор за минуту выполняет одну работу, а система из четырех процессоров делает это за две, то считается, что многопроцессорная система работает в два раза медленнее, однако имеет загрузку или пропускную способность в два раза больше. Оценка SPECrate дает представление об эффективности распределения ресурсов между процессорами, а также о способности операционной системы и компилятора строить разделяемый, параллельный код.

Набор тестовых программ для SPECrate аналогичен наборам SPECint92 и SPECfp92, однако эти программы размножаются на некоторое число одновременно запускаемых копий. Результирующее значение SPECrate вычисляется следующим образом:

```
SPECrate = число копий*ref_const*
n_cpu/общее время выполнения
```

Количество одновременно выполняемых задач может выбираться произвольным образом, при этом совсем не обязательно оно должно совпадать с количеством процессоров. Например, архитектура компьютера фирмы «Convex» Exemplar SPP предусматривает наличие высокопроизводительного канала между 128 процессорами, а также 32 Гб разделяемой памяти. Как будет организован процесс вычислений и где будут расположены задачи, зависит от каждой конкретной конфигурации. Величины ref_const и n_cpu для каждого конкретного теста являются постоянными коэффициентами. Общее время — это время завершения последней из всех запущенных работ.

В таблице 4 даны показатели SPECrate для некоторых систем стоимостью от 50 до 100 тыс. долл. в однопроцессорном варианте и до 200 тыс. долл. для конфигурации с двумя и более процессорами.

Оценка SPECrate хорошо характеризует возможности компьютера по организации многопользовательского режима для работы в системах коллективного пользования, а также комплексов обработки задач в реальном времени.

Храните деньги в надежном банке

Количество транзакций, выполняемых в единицу времени, позволяет оценить эффективность компьютера при решении задач информационного обеспечения — в работе с базами данных. Согласно определению, транзакция — это последовательность операций ввода/вывода, во время проведения которых база данных остается неизменной. Все происходящие изменения либо не видны, либо отсутствуют до тех пор, пока операция не завершится.

До 1988 года отсутствовали унифицированные методики оценки производительности компьютеров при выполнении транзакций. Только после создания Совета по проведению оценки скорости выполнения транзакций, получившего название TPC (Transaction Processing Performance Council), в который вошли такие ведущие фирмы — производители вычислительных систем, как «Hewllet-Packard», IBM и «Control Data», были установлены единые правила проведения измерений и оформления отчетов по их результатам. В настоящее время в качестве общепризнанного стандарта из комплекса TPC приняты оценки A, B и C.

TPC-A характеризует быстродействие выполнения транзакции в режиме on-line при работе кассира банка. Для выполнения данного

Таблица 3

Модель	SPECint92	SPECfp92
HP 712(PA-7100LC/80)	84,0	122,0
HP 735(PA-7150/125)	136,0	201,0
CDS 9140(R4400/150)	87,9	97,1
IBM 580(POWER PC/62.5)	73,3	134,6

Таблица 4

Модель	SPECrate_int92	SPECrate_fp92
HP 9000 T500	1888 (1 проц.*)	3563 (1 проц.)
HP 9000/800 H70	3757 (2 проц.)	7325 (2 проц.)
CDS 4680-312	2674 (2 проц.) 5300 (4 проц.)	2676 (2 проц.) 5523 (4 проц.)
SPARCserver 2000	2326 (2 проц.)	2953 (2 проц.)

*проц. — процессоры

теста специально эмулируется операционная обстановка банка (терминалы и линии коммуникаций), а в качестве транзакции выбирается обычная операция по обновлению счета клиента. Скорость работы в локальном окружении (без передачи транзакции во внешнюю сеть) измеряется в tps-A-local. Быстродействие при работе с внешними межбанковскими сетями оценивается в tps-A-wide. В отчет о проведенном испытании входит стоимость компьютера вместе с программным обеспечением и дополнительным оборудованием, необходимым для обеспечения работы банка в период до трех месяцев. Стоимость вычислительной системы включает также сопровождение за пять лет. Отношение общей стоимости комплекса к числу tps дает стоимость одной транзакции.

Оценка **TPC-B** является подмножеством TPC-A без эмуляции терминалов и линий связи и ориентирована на проверку возможностей только СУБД в условиях ее интенсивной эксплуатации. Единицами измерения являются tps-B и стоимость одной транзакции.

Тест **TPC-C** заключается в моделировании различных видов деятельности банка: операций со счетами, инвентаризации и так далее. Размер транзакций в TPC-C изменяется от очень простых и коротких до очень сложных и длинных операций, которые, как и в реальной практике бизнеса, требуют сложных проводок и многоступенчатых пересылок. Единицами измерения являются число транзакций в минуту (tpm) и стоимость одной транзакции.

Показатели по оценке TPC зависят не только от возможностей аппаратуры, но и от базы данных. Обычно применяются три СУБД: Oracle, Informix и Sybase. Для примера приведем таблицу 5 сравнения по TPC-B (цены компьютеров и необходимое программное обеспечение даны для США).

Значения оценок SPECrate и TPC находятся в непосредственной зависимости от количества процессоров в тестируемой компьютерной системе. Однако, всегда ли увеличение количества процессоров приводит к повышению общей производительности системы?

Общая производительность многопроцессорной системы изменяется от теоретически максимально возможной, равной производительности одного процессора, умноженной на число процессоров, до минимальной, когда суммарная производительность всех процессоров равна быстродействию одного. Если прикладные задачи пользователя являются принципиально последовательными (аналогичными тестам Li, Sc или Eqntott из набора SPECint92), то лучше использовать быструю однопроцессорную систему. Если выбираемая система будет предназначена для выполнения программ на языке Фортран для задач, аналогичных тестам Nasa7 или Tomcatv из набора SPECfp92, то увеличение числа процессоров может дать определенный выигрыш. Некоторую приблизительную оценку конкретного значения требуемого числа процессоров можно получить с помощью тестирования по оценке Linpack, одна из таблиц отчета по которому включает показатель эффективности загрузки процессоров в многопроцессорных системах. Результаты приведены в таблице 6, в которой t1 — время выполнения теста на одном процессоре, no_CPU — количество процессоров, а tn — время выполнения теста на указанном числе процессоров.

При числе процессоров больше шестнадцати в системе Intel iPSC, каждый из них основную часть времени ждет. Эффективность многопроцессорной системы определяется не столько быстродействием одного процессора, сколько пропускной способностью шины памяти и шин ввода/вывода, возможностями процессора при работе в мультипроцессорной конфигурации, а также способностью ОС распараллеливать выполнение задачи.

В общем случае, проблема выбора адекватного теста для оценки эффективности многопроцессорной системы является достаточно сложной задачей. Из рис. 1, показывающего на примере сервера CD4680 графики зависимости производительности от числа процессоров, видно, что каждый тест по разному характеризует эффективность организации многопроцессорных вычислений.

Решение задач, аналогичных используемым в тесте SPECrate, в большинстве случаев дает хорошие результаты при их выполнении на многопроцессорных системах. Оценка TPC-B, предполагающая интенсивную работу с базами данных, моделирует наиболее тяжелые для компьютера условия загрузки, которые в максимальной степени выявляют все недостатки многопроцессорной архитектуры. Как следствие, кривая эффективности расположена ближе к минимуму, и поэтому для банковских применений целесообразность использования нескольких процессоров требует дополнительного анализа.

В отличие от TPC оценки AIM, как будет показано далее, ориентированы на проверку компьютеров, работающих в более универсальных приложениях. Поэтому кривая эффективности для данной оценки обычно располагается между SPECrate и TPC. При наличии достаточных средств решение в пользу многопроцессор-

Таблица 5

Модель	Цена, тыс. долл.	TPC-B, tps-B	Цена транзакции, тыс. долл.	СУБД
HP9000/807S	106	41,07	2,583	Informix
InfoServer 4330	86	43,32	1,99	Informix
SPARCserver 2	100	39,7	2,541	Informix
IBM 520H	106	44,76	2,375	Informix

ного варианта в такой ситуации часто оказывается более эффективным.

Оценка архитектуры компьютера

Фирма AIM Technology специализируется на проведении оценок UNIX-систем и подготовке объективной информации о продукции различных фирм — производителей компьютеров. В отличие от других оценок тесты AIM специально ориентированы на получение интегральных характеристик системы. Тестированию подвергаются все компоненты компьютера при работе в многопользовательском и многозадачном режиме.

Методика **AIM** основана на проверке по следующим критериям:

- пиковая производительность (AIM Performance Rating) — максимальная производительность в режиме наиболее оптимального использования всех ресурсов компьютера;
- максимальная загрузка (Maximum User Load) — максимальное число заданий, которое может выполнить система за минуту при работе наибольшего количества пользователей;
- обработка утилит UNIX (Utilities Index или Milestone) — оценка возможностей по выполнению 40 утилит ОС UNIX;
- пропускная способность (Throughput Graph) — показатель производительности (количество работ в минуту) в зависимости от степени загрузки системы;
- цена (Price) — стоимость компьютерной системы.

Обзорные отчеты содержат следующие сравнительные диаграммы.

«Цена/Пиковая производительность» предназначена для выбора систем, достигающих своей пиковой производительности при работе в режиме нормальной эксплуатации при наименьшей цене.

«Цена/Максимальная загрузка» позволяет выбрать системы, достигающие своей пиковой производительности при работе в условиях максимальной загрузки за минимальную цену.

«Пиковая производительность» позволяет выбрать системы, достигающие наибольшей эффективности в режиме нормальной эксплуатации.

«Максимальная загрузка» позволяет выделить системы, демонстрирующие наилучшие показатели при работе с максимальной загрузкой.

«Производительность при выполнении UNIX-утилит» идентифицирует системы, наиболее эффективно выполняющие утилиты ОС UNIX за одну минуту.

В каждую диаграмму включается семь компьютеров, показавших наилучшие результаты в своей ценовой группе: до 15 тыс. долл., 15 — 25 тыс. долл., 25 — 50 тыс. долл., 50 — 100 тыс. долл., 100 — 500 тыс. долл. и выше 500 тыс. долл.

Основным набором оценок собственно компьютерной системы,

Оценка многопроцессорных систем по тесту Linpack 1000 × 1000 *Таблица 6*

Модель	t1	no_CPU	tn	Эффективность
DEC 10000 Alpha	4,31	4	1,28	0,84
DEC 10000 Alpha	4,31	2	2,44	0,88
IBM 3090/xxxS	7,27	4	1,89	0,96
IBM 3090/xxxS	7,27	2	3,64	0,99
Intel iPSC/860	22	16	5,1	0,26
Intel iPSC/860	22	4	8,9	0,6
Intel iPSC/860	22	2	12,8	0,84
Sun SPARC2000/50	23,85	8	2,99	0,99
Sun SPARC10/514	24,73	4	6,81	0,91
DEC VAX 62xx	1295	4	332	0,98
DEC VAX 62xx	1295	2	654	0,99

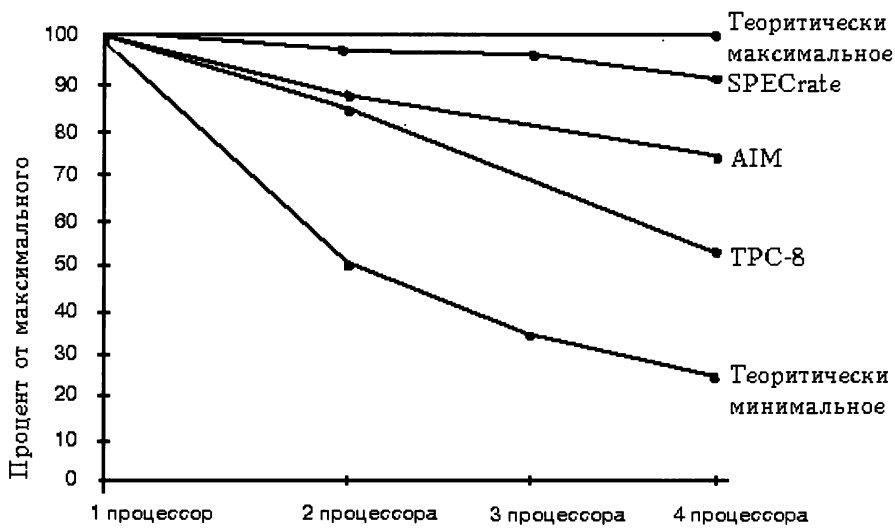


Рис. 1. Производительность сервера CD 4680

без вывода на терминалы, учета производительности при работе с X Windows и в составе сети, является AIM System Benchmark (Suite III). Набор состоит из шести моделей: обмены с оперативной памятью (20%), работа с вещественными числами двойной и одинарной точности (10%), операции с целыми числами (20%), обмены данными между процессорами (10%), вызовы функций на языке Си с 0, 1, 2 и 15 параметрами (20%), ввод/вывод на диск (20%).

В таблице 7 приведены данные из отчета AIM на осень 1994 г.

Если сравнить эту таблицу с таблицей 8, содержащей данные из отчета AIM за осень 1993 г., то можно заметить существенные изменения.

За прошедший год появились новые модели и значительно сократился разрыв значений цена/производительность. Фирма «Hewlett-Packard» за счет использования более производительного процессора сильно укрепила шансы модели 715, а модель 9130 и ее разновидность 913GC фирмы «Control Data» также сильно изменили расстановку сил в данном классе компьютеров. Однако настоящая революция произошла в классе недорогих систем, стоимостью до 15 тыс. долл., где фирма HP лидирует со своей рабочей станцией 712. Неплохие позиции в этом классе занимают и модели серии Altos фирмы «Acer» — Altos 700, сервер рабочей группы, и Altos 7000, сервер среднего уровня, опережая по показателю цена/производительность модели DEC 300L и SGI Indy.

После диаграмм сравнения в обзорном отчете размещаются данные по компьютерам в отдельности — это краткое описание, включающее только основные параметры каждой модели (рис. 2).

Полный отчет по конкретной компьютерной системе содержит еще три страницы с данными тестирования по набору тестов AIM Subsystem Benchmark (Suite II). Проверка по данному набору производится при работе компьютера в однозадачном режиме и включает следующие оценки (рис. 3):

1. Оценка производительности при работе с диском (килобайты в секунду) применяется при выборе систем для работы с базами данных, файловых серверов и рабочих мест разработчика программного обеспечения.
2. Производительность выполнения операций над вещественными числами, измеряемая в тысячах операций в секунду отдельно для сложения, умножения и деления, используется для оценки систем для работы в научных и физических приложениях.
3. Производительность при работе с целыми числами. Измеряется в тысячах операций в секунду отдельно для сложения, умножения и деления чисел в длинном (long) и коротком (short int) формате. Данная оценка используется при выборе систем для работы в финансовых приложениях.
4. Производительность операций чтения/записи в память. Измеряется в килобайтах в секунду отдельно при чтении и записи для целых чисел и символов. Данная оценка используется при выборе компьютеров для работы в издательских системах и в финансовых приложениях.
5. Производительность операций копирования в памяти. Измеряется в килобайтах в секунду при копировании целых чисел в длинном и коротком формате, а также символов.

Оценка «Цена/пиковая производительность» для группы 15-25 тыс. долл. Осень 1994 года. *Таблица 7*

(Выборка из 29 систем)

Модель	Цена/AIM
Zenit Z-Server EX	214
Hewlett-Packard 9000 Model 715-100	218
SGI Indigo R4000 (32 Мб)	232
Hewlett-Packard 9000 Model 715-75	246
Olivetti LSX 5050/2	268
Control Data 9130 R4000	275
ICL E240	280

Оценка «Цена/Пиковая производительность» для группы 15-25 тыс. долл. Осень 1993 года. *Таблица 8*

(Выборка из 29 систем)

Модель	Цена/AIM
SGI Indigo R4000 (32 Мб)	232
SGI Indigo R4000 (96 Мб)	312
Motorola Series 900 Model M921	349
Zenith Data Systems Z-Server EX P60E 1000A	385
Wyse Series 7000i Model 760MP (2 процессора)	398
Sun SPARCServer 10 Model 40	401
DECsystem 5000 Model 50	408

6. Производительность операций в памяти над массивами ссылок. Измеряется в тысячах ссылок в секунду для целых чисел в длинном и коротком формате.
7. Производительность при вызове системных функций. Измеряется количеством обращений в секунду к таким функциям UNIX, как create/close, fork, sbrk(), signal и unmask().
8. Производительность при вызове функций в прикладной задаче. Измеряется количеством вызовов в секунду для функций без параметров, функций с одним, двумя и пятнадцатью integer параметрами.

От общего к частному

Приведенные оценки позволяют получить общее представление о возможностях компьютера, однако могут так и оставить без ответа вопрос, который задают себе многие потенциальные покупатели: «А все-таки как данная конфигурация проявит себя при решении именно моих, конкретных задач?». Частично на этот вопрос отвечает дополнительная глава «Производительность рабочих станций», которую фирма AIM размещает в своем итоговом отчете с весны 1994 года.

Глава содержит оценки измерения производительности станций при выполнении смеси задач из различных прикладных областей: механические САПР и инженерный анализ, геоинформационные системы и обработка изображений, банковское дело. В ходе тестирования измеряется пропускная способность системы, скорость выполнения ввода/вывода, а также выполнение функций UNIX. Например, при выполнении специализированного тестового набора, моделирующего бизнес-приложения, показатели моделей HP 715 и SunClassic равны соответственно 44,6 и 35,5 работ/минуту. Максимальная загрузка равна соответственно 43,6 и 34,5 AIM.

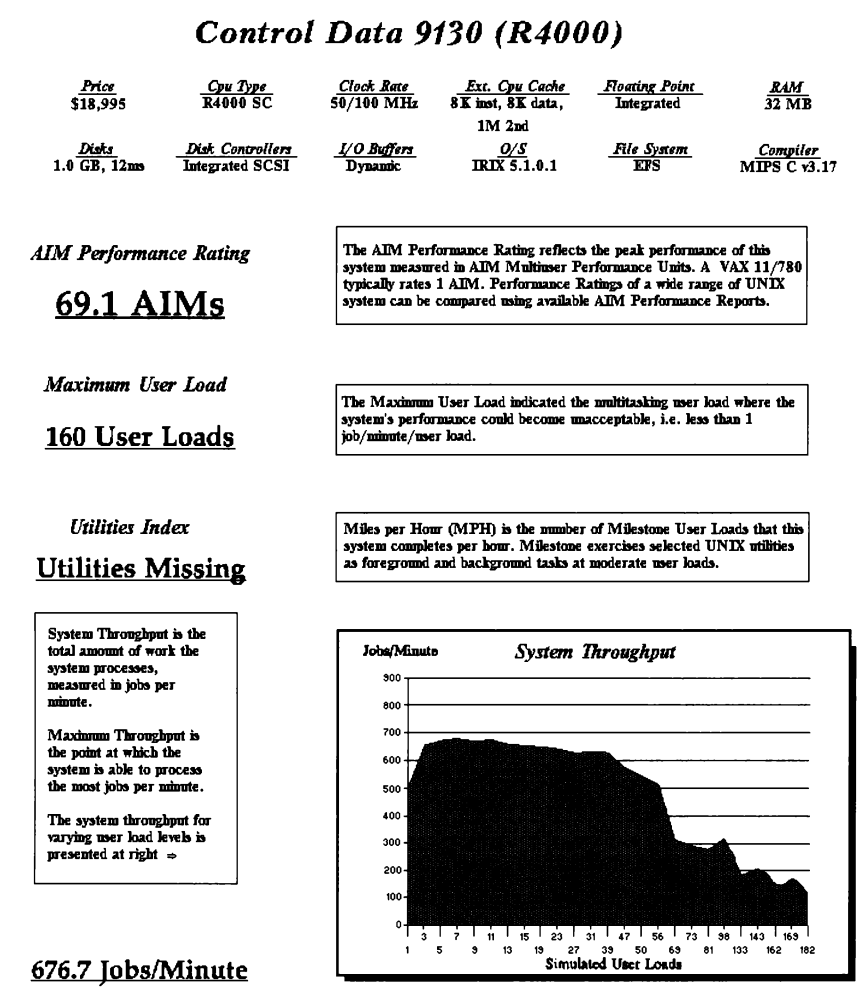


Рис. 2. Пример страницы отчета AIM Subsystem

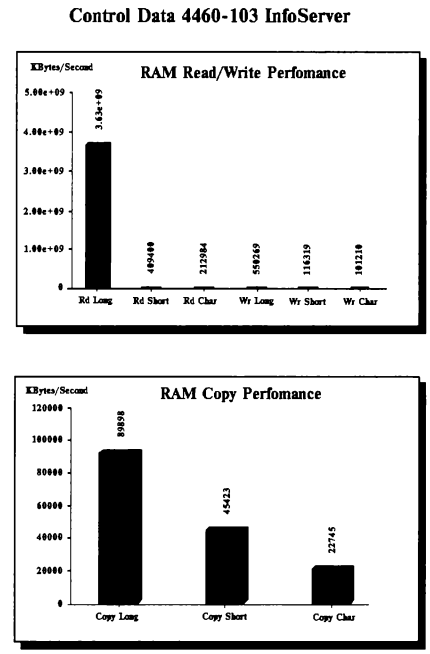


Рис. 3. Пример страницы отчета AIM Subsystem Benchmark

Примером более детального тестирования производительности при выполнении конкретных прикладных задач могут быть более специализированные тесты, такие как CADbench, предложенный фирмой «Intergraph».

Оценка CADbench93 предлагается в качестве основы для формирования различных тестов проверки производительности аппаратных платформ, версий и конфигураций системного программного обеспечения при решении задач САПР. Сегодня CADbench функционирует в среде MicroStation и AutoCAD, а благодаря тому, что в основу теста были положены системы программирования MDL и ADS, его можно адаптировать для любой аппаратной платформы, поддерживающей обе системы.

Проверка в тесте CADbench осуществляется по трем показателям: CADvec — перерисовка векторов , CADrender — визуализация и CADio — ввод/вывод. Общее время,

необходимое для выполнения полного комплекта тестов, нормализуется относительно стандартной аппаратной платформы, например Gateway 486/33.

В состав CADvec входят операции перерисовки всего экрана, вывод окна, центрирование окна, заливка, высвечивание и затемнение всех слоев чертежа, поворот на угол 45 градусов. Тест CADio состоит только из одной операции — открытие файла. Операции сглаживания кривой и удаления невидимых линий образуют тест CADrender. В таблице 9 приведены некоторые сравнительные данные для моделей HP 712/80, SGI Indy, TD2 фирмы «Intergraph» и SPARC 10/51. Все рабочие станции принадлежат примерно одной ценовой группе. Сравнение осуществляется относительно модели Gateway 2000, производительность которой взята за 100.

Компьютеры TD2 (процессор Intel Pentium, 512 кб внешний кэш, 32-256 Мб оперативная память, пять PCI слотов, один PCI/EISA, два EISA, встроенный PCI 10BaseT–адаптер, контроллер PCI Fast SCSI II и один внутренний 3,5" SCSI диск) демонстрируют наибольшие показатели производительности при использовании именно Windows NT. Из таблицы видно, что система TD2 на тесте CADbench93 работает в 1,7 раза быстрее, чем станция SPARC 10/51, что лишний раз свидетельствует о том, что в каждом конкретном случае при выборе конфигурации и модели необходим тщательный анализ. Персональный компьютер может с успехом быть рабочей станцией, а рабочая станция, напротив, может выполнять роль персонального компьютера.

Любая модель является лишь грубым приближением реальной жизни, поэтому при всем своем многообразии, перечисленные тестовые оценки могут не дать полной картины поведения конкретной вычислительной системы при решении реальной задачи. Очень часто многое зависит от конфигурации, выбранной для тестирования. Можно привести множество примеров,

Таблица 9

Модель	HP 712/80	Indy	TD2	SPARC 10/51
ОС	NT v.5	NT v.5	NT v.5	Solaris 2.2
CADio	27,33	35,85	29,90	39,17
CADvec	45,45	62,18	35,95	64,43
CADrender	63,03	77,80	51,50	70,65
CADbench	313	254	418	245

когда при проведении самостоятельной оценки, фирма-производитель выбирает для сравнения модели, неадекватные по конфигурации. Анализируя материалы, особенно если это данные по новым изделиям, полезно обращать внимание на часто очень невыразительные сноски, предупреждающие о том, что цифры получены самой фирмой-производителем. Наиболее надежный, но возможно не самый быстрый способ — это сравнение по оценкам, полученным в результате проведения стандартизированных тестов независимыми организациями. В любом случае, требуется аккуратно сопоставлять параметры конкретных аппаратных и программных конфигураций, выбираемых для сравнения: частоту процессора, размер кэш-памяти, объем оперативной памяти, версию ОС и компилятора.

Особое внимание следует уделять стоимости конфигурации, при которой достигаются рекордные результаты. Например, промелькнувшее в «ComputerWorld — Moscow» сообщение «Производительность модели HP9000/735 аналогична быстродействию станции SGI Onyx при цене в два раза меньше» не несет в себе ровно никакой информации, а только окончательно запутывает читателя. Из оценок AIM видно, что показатели во многом зависят от памяти, типа процессора и емкости внешней памяти. В станции Onyx используются процессоры R4400/150 МГц, а конкретная конфигурация памяти может изменяться от 32 до 384 Мб. Данный компьютер изначально комплектуется графическими процессорами VTX или Reality Engine (192 битовых плоскости, 32 бит Z-буфер, 1,6 млн. Tmesh/с и все это на экране 19 дюймов), которые по цене действительно превышают CRX-48Z. Однако если еще упомянуть возможности VTX по обработке текстур, то станет очевидным, что это уже принципиально другой, более высокий класс машин. Корректнее сравнивать HP 735 с моделью Indigo2 Extreme. Интегральные оценки для HP 735 и модели Indigo2, которая намного уступает Onyx, равны соответственно 71,7 и 72 при ценах 41 и 23 тыс. долл. Графические возможности процессоров CRX-48Z и Extreme почти одинаковы. В результате остается двукратное превышение производительности процессора по тесту Linpack (41 и 26), однако этого явно недостаточно для вывода об интегральном двукратном превышении производительности.

Сравнение по различным оценкам позволяет получить только первое приближение, а именно: из 10-15 моделей разных фирм выбрать 3-4 наиболее удачных по конфигурации. Если решение об использовании рабочих станции принято всерьез и надолго, то вторым приближением будут ваши собственные тесты, максимально отражающие реальные потребности конкретного приложения. Уважающие себя фирмы-интеграторы могут предоставить компьютерную систему во временное пользование, что даст вам возможность оценить эффективность решения собственных задач. ■

Вы могли бы и больше заплатить за сервер, но придется ограничиться меньшими расходами

ALPHA™ GENERATION

Новое 64-разрядное семейство AlphaServer сегодня, если Вам нужна высочайшая производительность 64-разрядных RISC-процессоров, непревзойденные возможности наращивания вычислительной мощности, исключительная надежность и лучшие гарантии для серверов в промышленности, Вы вполне можете получить это без чрезмерных затрат, если купите одну из удивительных моделей нашего нового семейства AlphaServer. Новые серверы поставляются с обширным программным обеспечением — свыше 6000 прикладных программ UNIX®, Windows NT™ и Open VMS™. Они предусматривают установку дополнительных аппаратных средств стандартной для промышленности локальной шины PCI, имеют увеличенную пропускную способность системы ввода-вывода, больше оперативной и больше внутренней массовой памяти. Фактически единственное, что не увеличено, это цена.

Никто и нигде не предложит Вам столь выгодной покупки!

	AlphaServer 1000 4/200	AlphaServer 2000 4/200	AlphaServer 2100 4/275
Число процессоров (SMP)	1	1 – 2	1 – 4
TPS (транзакция/с)	до 285	до 400	до 850
SPECint 92	135,8	126,7	200,1
Макс. показатель SPEC...92	3135	5778	15470
Число гнезд ВВ	2PCI,7EISA 1комбинир.	3PCI,1EISA	3PCI, 8EISA
Макс. пропускная способность ВВ	132 Мбайт/с	132 Мбайт/с	132 Мбайт/с
Макс. емкость ОЗУ	512 Мбайт	640 Мбайт	2 Гбайт
Макс. емкость внутренней массовой памяти	14 Гбайт	16 Гбайт	32 Гбайт

digital™

Представительства:
в Москве тел. (095) 244-9540
в С.-Петербурге тел. (812) 119-1040
в Киеве тел.: (044) 295-6910, 295-1431. Факс: (044) 293-6041

Партнер, которому отдают предпочтение

Сегодня мы продолжим рассказ о направлениях работы фирмы **«Digital Equipment Corporation» (DEC)**, начатый в «Мониторе» № 7 и 8 за 1994 год.

Широкому кругу специалистов и пользователей компьютеров компания DEC известна прежде всего как производитель рабочих станций и серверов, отличающихся наивысшей производительностью и прекрасным соотношением «цена/производительность» среди машин соответствующего класса. Хочется отметить комплексный подход DEC к разработкам: в основе всего — микропроцессоры семейства **Alpha AXP**, но их мощь поддерживается успехами «на всех фронтах». Можно назвать семейство графических ускорителей **ZLX-E**, обеспечивающих великолепную производительность двумерной и трехмерной графики, контроллер **HSD30** дисковых массивов подсистем DSSI, дающий возможность подключить к одному DSSI порту до двадцати устройств, дисковых массивов с RAID структурой уровня 0, **кэш-память с обратной записью** для контроллеров серии HSJ, HSZ40 и HSD30, **ленточные и оптические накопители**, позволяющие сохранять уже не гигабайты, а терабайты информации.

Но и это не все. Руководствуясь принципом комплексности, фирма «Digital» предлагает потребителям **законченные аппаратно-программные решения**.

Статья подготовлена по материалам, предоставленным фирмой DEC. Публикуется на коммерческой основе.

«Мультиклиент» — средство доступа к любой информации в сети

Корпорация DEC выпустила **Multia Desktop** — единый клиент, обеспечивающий удобный высокопроизводительный «прозрачный» доступ к Microsoft Windows, Windows NT, UNIX и существующим приложениям host компьютеров в любой точке сети. Multia Desktop работает в среде «клиент/сервер» и обеспечивает доступ к любым источникам информации в сети нажатием клавиши «мыши».

Среда Multia Desktop настраивается в соответствии с требованиями пользователя, так как программное обеспечение имеет возможности управления конфигурацией и приложениями. Например, пользователь щелчком «мыши» может запустить программу или обратиться к помощи средств автостарта для автоматического запуска приложений, вне зависимости от того, где они находятся.

Multia Desktop работает под управлением операционной системы *Microsoft Windows NT Workstation 3.5* и поддерживает многие сетевые протоколы: *IPX/SPX, LAN Manager, NetBIOS, PATHWORKS, TCP/IP и DECnet*. Это дает возможность вызывать самые разные приложения:

- локально могут выполняться приложения *Win16* и *Win32*, загруженные по сети с файловых серверов;
- *X-приложения* любой системы, базирующейся на UNIX или OpenVMS, функционируют благодаря встроенному серверу X11.R6;
- *приложения host компьютеров* позволяет загрузить интегрированный эмулятор видеотерминала Attachmate KEA!340.

Необходимо отметить, что платформой Multia Desktop является микропроцессор *Alpha AXP 166 МГц*, а среди технических характеристик можно назвать *базовую память размером 24 Мб*, расширяемую до 256 Мб, и стандартные *слоты PCMCIA и PCT*, что, наряду с высокопроизводительным графическим ускорителем и встроенным аудиооборудованием, обеспечивает технологии мультимедиа, FDDI и различные сетевые соединения.

По словам *Ларри Кабринети*, вице-президента подразделения комплектующих и периферийных изделий «Digital», «на основе фундаментальных требований развивающейся архитектуры «клиент/сервер» «Digital» разработала первое в мире «мультиклиентское» рабочее место и заново определила понятие «клиент» в вычислениях «клиент/сервер»».

Новые решения для систем накопителей данных

«Digital Equipment Corporation» не только выпустила новые дисковые и ленточные устройства, но и предложила *решения для платформ Alpha*

AXP и VAX с операционными системами Open VMS и OSF/1.

Программные средства управления системами накопителей данных призваны существенно увеличить производительность администраторов и пользователей.

- **Hierarchial Storage Management for Open VMS V1.0** — программный продукт, который работает на нескольких уровнях, расширяет файловую систему Open VMS и позволяет эффективно управлять доступом к данным, сокращая время архивации.
- **Save SetManager for Open VMS V1.0** — программный продукт, существенно сокращающий время выполнения функций проверки, копирования и слияния, работает на нескольких уровнях.
- **Desktop Backup/Archiver V2.0** — средство нового поколения для управления архивацией в LAN среде «клиент/сервер» системы PATHWORKS предоставляет возможности копирования, восстановления и архивации данных. Позволяет сохранять данные с компьютеров IBM PC и Macintosh на серверах Open VMS VAX и Alpha AXP.
- **Optical Storage Management Software V3.2** обеспечивает использование оптических дисков в среде Open VMS.
- **Networker Save and Restore Software (NSR) V3.0** является надежным средством защиты данных в неоднородных сетях TCP/IP, включает новый элемент Media pool — мощное средство для управления архивами нарастающих объемов. NSR поддерживает возможность архивации для двадцати различных платформ на серверах OSF.
- **Storage Library System (SLS) V2.4** для платформ Open VMS VAX и Alpha AXP — средство управления процессами сохранения и архивации важнейших корпоративных данных. SLS работает на нескольких уровнях и предназначается для работы с наборами сменных накопителей и устройств.

Среди новых решений для различных накопителей можно отметить **StorageWorks** — семейство подсистем сохранения данных, включающих в себя дисковые, ленточные устройства и дисководы оптических дисков. Используя этот продукт, можно создавать модульные конкурентоспособные решения. StorageWorks применяется в разных операционных системах и на разных аппаратных платформах, включая системы Alpha AXP, VAX, среду Novell, рабочие станции фирмы «SUN Microsystems».

Эд Мак-Каррен (Ed McCarren), менеджер фирмы DEC по маркетингу программных средств управления системами накопителей данных, говорит: «Мы удовлетворены нашим все более расширяющимся семейством продуктов для управления процессами сохранения данных. Совмещая эффективные накопители большой емкости с мощным программным обеспечением, мы предоставляем клиентам комплексные решения, полностью обеспечивающие управление процессами сохранения данных от отдельно взятых рабочих мест до целого предприятия.»

Семейство систем Alpha расширяется

Фирма DEC выпустила в продажу **3 новых модели рабочих станций** в ценовой категории от $10000 до $15000:

- настольная **AlphaStation 200 4/166**,
- настольная **AlphaStation 200 4/233**,
- напольная (mini-tower) **AlphaStation 400 4/233**.

Но это еще не все! Вместе с ними на рынке появились **5 вариантов серверов**:

- серверы низкого уровня **AlphaServer 1000 4/200**,
- многопроцессорные серверы низкого уровня **AlphaServer 2000 4/200**,
- SMP серверы низкого уровня для отделений **AlphaServer 2100 4/275**,
- SMP серверы высокого уровня для отделений (кабинетная версия) **AlphaServer 2100 4/275**,
- многопроцессорные серверы уровня предприятия **DEC 7000 Model 700**.

Видимо, внимательный читатель заметил, что появились *новые торговые марки AlphaServer и AlphaStation*; новое название AlphaServer 2000 4/200 введено для пользующегося повышенным спросом сервера Digital 2100 A500MP.

Альтернатива mainframe

Хотелось бы несколько подробнее остановиться на многопроцессорном сервере **DEC 7000 Model 700**. *Паулина Нист* (Pauline Nist), вице-президент подразделения серверов «Alpha Systems Business Group», так оценивает его: «Предлагаемая нами альтернатива mainframe — система DEC 7000 Model 700 с 6 процессорами — превосходит по производительности HP9000 Model T500 с 12 процессорами.» Эта модель является мощнейшей среди серверов для предприятия и победителем по показателю общей производительности в соревнованиях *«AIM Technology's 1994 Hot Iron Awards»* для компьютеров с UNIX-системами. Она базируется на Alpha AXP 21064A 275 МГц, оснащена шестью симметричными мультипроцессорами и обладает производительностью *1350 транзакций в секунду*.

Ценность этой модели, как и других серверов и рабочих станций, заключается в том, что они *совмещают производительность и широкий спектр применения*, характерные для больших машин, *с низкими затратами на обновление компьютера*, что во всех отношениях лучше для клиента.

Настраиваемая платформа фирмы «Digital» Alpha AXP, современные системы хранения информации и сетевая поддержка, развитое программное обеспечение и высококлассный сервис вместе с ориентацией на конкретные отрасли промышленности своих бизнес-партнеров помогают DEC побеждать в конкурентной борьбе на современных международных рынках.

«На сегодняшний день мы установили конкурентоспособную систему скидок, упростили условия и положения контрактов по программному обеспечению, проводили предварительные продажи и маркетинговое изучение анонсируемых продуктов, — сказал *Гарольд Копперман* (Harold D. Copperman), вице-президент по глобальным продажам и маркетингу подразделения бизнес-систем фирмы «Digital». — Фирма «Digital» намеревается стать «партнером, которому отдают предпочтение». Мы разрабатываем программы лидерства, чтобы осуществить это обязательство.» ■

COMTEK

COMTEK '95 *24-28 апреля*

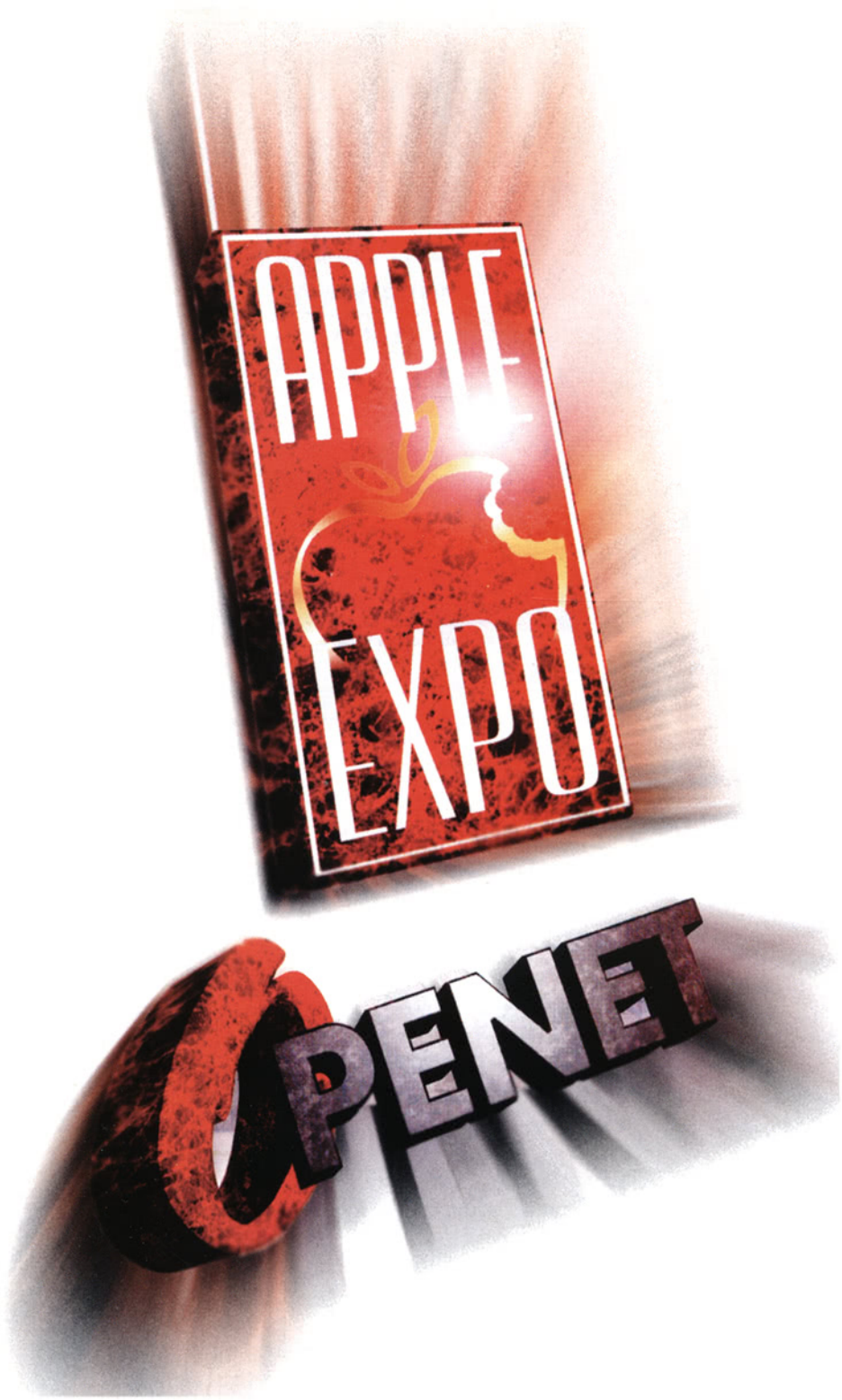
Организаторы выставки:

C·O·M T E K INTERNATIONAL

CROCUS International

6 - Я МЕЖДУНАРОДНАЯ СПЕЦИАЛИЗИРОВАННАЯ ВЫСТАВКА КОМПЬЮТЕРНОЙ ТЕХНИКИ, КОММУНИКАЦИЙ И КОНФЕРЕНЦИЙ.

Выставка расположена в павильонах № 2,3в 1/2павильона"Форум".



APPLE EXPO

OPENET

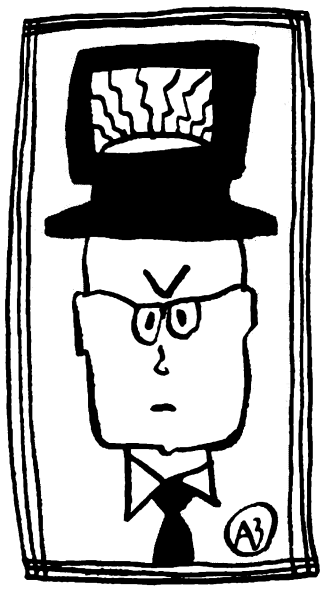
COMTEK INTERNATIONAL

CROCUS International

На выставке "КОМТЕК '95" развернутся экспозиции **OPENET** и **APPLE EXPO**.

Выставки расположены:
OPENET в павильоне № 1. APPLEEXPO в 1/2 павильона "Форум".

Справки,
дополнительная информация.
Телефон: (095) 249-86-06,
факс: (095) 249-86-09.



Нейронные сети на пути к российскому рынку

А. Масалович

Невероятно, но факт: на перенасыщенном компьютерном рынке Москвы, где продавцы сражаются за каждый доллар, полностью отсутствует один вид компьютерных изделий. Более того, не какой-нибудь экзотики, а массовых изделий, которые на Западе можно найти в каждом банке и в каждом танке. Годовой объем продаж этих изделий оценивается более чем в 600 млн. долларов, а темпы роста соответствующего рынка уступают лишь индустрии сотовой связи. Речь идет о нейроплатах и нейрокомпьютерах — может быть, единственном виде электроники, на котором «новые русские» еще не сделали своих первых миллионов.

Позвольте предложить вам маленький тест. Финансовая корпорация American Express, джинсовая империя Levi Strauss, машиностроительный концерн Sumitomo, сеть универмагов J.C.Penney, Bank of America — что общего между этими, столь разными на первый взгляд, компаниями? Правильно, все они входят в число наиболее преуспевающих компаний мира и присутствуют в престижном списке Fortune Top 100. Но объединяет их не только это. Все перечисленные фирмы используют в своей деятельности нейрокомпьютеры — последний «писк» западной компьютерной индустрии. И они не одиноки. Сегодня нейрокомпьютеры и нейроплаты активно применяют военные и медики, финансовые брокеры и налоговые инспекторы, специалисты по обработке изображений и многие, многие другие.

Таможенники используют нейроплаты для обнаружения пластиковых бомб и наркотиков, финансисты — для предсказания колебаний курсов валют, летчики — для распознавания наземных целей, банкиры — для оценки кредитных рисков. В тех случаях, когда задача не поддается формализации, а входные данные неполны, зашумлены и противоречивы, применение традиционных компьютеров становится неэффективным. И тут на помощь приходят нейрокомпьютеры, обладающие способностью самообучаться на примерах и настраиваться на решение определенных классов задач. Итог — сфера применения нейрокомпьютеров охватывает сегодня все основные области человеческой деятельности, а объем рынка нейросетевого оборудования удваивается каждые два года.

Чтобы объяснить, что такое нейроплата, нейрокомпьютер и нейронная сеть, необходимо сделать небольшой экскурс в историю. В 1943 году У. Маккалоу (U. Makkaloh) и У. Питтс (W. Pitts) сформулировали основные положения теории деятельности головного мозга и разработали математическую модель нейрона. В представлении Маккалоу-Питтса мозг представляет собой ансамбль нейронов, имеющих одинаковую структуру. Каждый нейрон реализует некоторую функцию, называемую пороговой, над входными значениями. Если значение функции превышает определенную величину-порог (что характеризует суммарную значимость полученной нейроном информации), нейрон возбуждается и формирует выходной сигнал для передачи его другим нейронам. Вот, собственно, и все. Пройдя путь от рецепторов (слуховых, зрительных и других) через нейронные структуры мозга до исполнительных органов, входная информация таким образом преобразуется в набор управляющих воздействий, адекватных ситуации.

Принципы, изложенные в работах Маккалоу-Питтса, были в 1958 году воплощены Ф. Розенблаттом (F. Rosenblatt) в оригинальном устройстве — персептроне «Марк-I», способном сносно распознавать печатные буквы и обучаться на примерах. Персептрон заинтересовал военных (еще бы — в распознавании нуждаются не только печатные тексты, но и ракеты противника, например), и следующее десятилетие ознаменовалось активными разработками в области нейронных сетей и их технических воплощений. Однако довольно быстро разработчики стали замечать, что применяемая нейросетевая парадигма (включающая в себя вид входных данных, пороговой функции,

структуры сети и алгоритмов обучения) несет в себе ряд существенных ограничений и не позволяет добиваться приемлемого качества при решении многих практических задач. Так, персептрон весьма чувствителен к поворотам образцов, предъявляемых для распознавания. В 1969 году вышла книга М. Минского (M. Minsky), доказавшего, что ограничения персептрона принципиальны и непреодолимы — и нейронные сети были оттеснены на задворки науки почти на двадцать лет. И только в последнее десятилетие, благодаря революционным работам Хопфилда (D. Hopfield) и Хехт-Нильсена (R. Hecht-Nielsen), теория нейронных сетей обрела вторую жизнь.

Сохранив основные черты базовой парадигмы Маккалоу-Питтса, Хопфилд усложнил структуру сети (главное — сделал ее многослойной), разработал новые алгоритмы обучения и доказал, что в таком модифицированном виде сеть способна решать принципиально более сложные задачи, нежели персептрон Розенблатта. Сегодня известно уже более десяти различных парадигм нейронных сетей (в том числе не только детерминированных, но и вероятностных), они реализованы в специализированных кристаллах и платах, на их основе созданы мощные рабочие станции и даже суперкомпьютеры.

Каким образом нейронная сеть решает задачи? Как вы знаете, для решения неформализованных задач традиционно применяются два основных подхода. Первый, основанный на правилах (так называемый rule-based), характерен для экспертных систем. Он базируется на описании предметной области в виде набора правил «если-то» (аксиом) и правил вывода. Искомое знание представляется в этом случае теоремой, истинность которой доказывается посредством построения цепочки вывода. При этом подходе вам, увы, необходимо заранее знать весь набор закономерностей, описывающих предметную область.

Совершенно по-иному обстоит дело при использовании другого подхода (так называемого case-based), основанного на примерах. В этом случае от вас не требуется знания каких бы то ни было правил — надо лишь иметь достаточное количество примеров для настройки адаптивной системы с заданной степенью достоверности. Нейронные сети представляют собой классический пример технологии, основанной на примерах. Процесс обучения сети новому классу задач включает следующие стадии:

АППАРАТУРА

Настроенная и обученная сеть будет не только подсказывать вам решения, но и оценивать степень их достоверности.

1. Формулируется постановка задачи (например, предсказание краткосрочных колебаний курсов ценных бумаг) и выделяется набор ключевых параметров, характеризующих предметную область. Обычно для этого привлекаются эксперты по соответствующей проблеме, которые перечисляют набор факторов и данных, учитываемых ими при принятии решения. Заметим, что от эксперта не требуется знания взаимозависимостей между входными данными и даже оценок их сравнительной важности.

2. Выбирается парадигма нейронной сети, наиболее подходящая для решения данного класса задач. Как правило, современные нейроплаты и нейрокомпьютеры позволяют реализовать не одну, а несколько базовых парадигм. Что касается критериев выбора базовой модели — на эту тему уже написаны многочисленные тома, где проведен первичный анализ всех мыслимых задач, будь то идентификация подписи, контроль качества питьевой воды или предсказание социальных взрывов.

3. Подготавливается возможно более широкий набор обучающих примеров, организованных в виде векторов входных параметров, ассоциированных с известными выходными значениями. Для задачи с предсказанием курсов валют, например, это значения всех курсов на данной бирже за определенное время плюс набор параметров, характеризующих общую биржевую и экономическую ситуацию, плюс данные о положении на других биржах на тот же момент. В качестве выходного эталона выбирается тот параметр, значение которого вы собираетесь предсказывать в будущем (например, курс доллар/иена). Входные вектора для обучения могут быть неполны и частично противоречивы — главное, что от них требуется, это максимальная представительность. Кстати, на Западе базы данных для настройки нейроплат на конкретную прикладную область стоят зачастую дороже самих плат.

4. Входные вектора по очереди предъявляются нейронной сети, а полученное выходное значение сравнивается с эталоном. Далее прослеживается путь от выходов сети до ее входного слоя, и все нейронные межсоединения, способствовавшие принятию правильного решения, поощряются — их веса увеличиваются. Те же пути, которые привели к ошибкам в компонентах выходного вектора, наказываются уменьшением веса (как правило, в зависимости от величины ошибки). Не правда ли, это напоминает процесс отбора и обучения советников какого-либо политического лидера?

5. Процесс повторяется до тех пор, пока суммарная ошибка в реакции сети на всех предъявленных примерах не станет меньше наперед заданной величины, либо пока сеть не придет в стационарное состояние. Рассмотренный алгоритм обучения носит название «обратного прохождения» (back-propagation) и относится к числу классических алгоритмов нейроматематики. Сегодня, кстати, известны гораздо более мощные и быстрые алгоритмы обучения.

6. Настроенную и обученную сеть вы можете смело использовать, предъявляя ей реальные ситуации.

Сеть будет не только подсказывать вам решения (а выходной вектор будет сформирован в любом случае), но и оценивать степень их достоверности. Кстати, по желанию пользователя, настройку сети можно совершенствовать и в процессе практической эксплуатации.

Теперь необходимо сказать о реализации нейронных сетей в электронных изделиях. В настоящее время существует целый ряд фирм (главным образом, в США и Японии), специализирующихся на нейросетевых продуктах. Наиболее массовым продуктом являются программные пакеты, моделирующие нейронную сеть непосредственно на ПК, а также нейроплаты — акселераторы для ПК и рабочих станций.

Из всего многообразия нейропакетов и нейроплат, представленных на мировом рынке, для использования в России наиболее интересны продукты семейства BrainMaker, распространяемые американской фирмой California Scientific Software. Причина не в том, что они лучше, быстрее или удобнее других. Нет, просто они дешевле, а для освоения нового рынка это немаловажно. С помощью BrainMaker можно преодолеть барьер вхождения в новую область менее чем за 1000 долларов — столько стоит базовая версия BrainMaker Neural Network Software v.3.1 для DOS или Windows. Профессиональная версия, содержащая генетическую обучающую систему и расширенные средства редактирования, стоит менее 2000 долларов. Отдельно поставляются специализированные базы данных для настройки сети на конкретные предметные области (в основном, банковские, биржевые, кредитные). Цена таких баз данных, как правило, составляет около 500 долларов.

Важным преимуществом пакета BrainMaker является возможность использования его совместно со специализированным аппаратным акселератором. Это нейроплата на базе сигнальных процессоров TI TMS320C30, обеспечивающая 4-кратное повышение скорости решения нейросетевых задач по сравнению с 486-DX2/66. Нейроплаты для BrainMaker стоят от 4000 долларов (что выгодно отличает их от продуктов других фирм, которые обычно не опускают планку ниже 10000 долларов).

К сожалению, сроки поставки даже самых простых нейросетевых продуктов составляют 3-6 недель. Нейронные сети — изделия стратегические, и даже после безвременной кончины COCOM для их поставок необходима лицензия.

Те, кто собирается самостоятельно заниматься разработкой нейросетевых продуктов, останавливают свой выбор на другом классе изделий — так называемых нейрочипах, где нейросетевая парадигма реализована в одном заказном кристалле. Наиболее мощный набор средств разработчика предлагает фирма «Intel». В него входят нейрочипы семейства i80170_CHIPSET (около 1500 долларов за штуку), нейроплаты EMB Multi-Chip Board (12000 долларов) и интегрированная система для разработки приложений (около 15000 долларов). Наиболее быстрые нейроБИС выпускает фирма Adaptive Solutions (1,2 млрд. соединений/с, 262 тыс. синапсов), наиболее дешевые — фирма Micro Devices (кристалл MD1220, моделирующий 8 нейронов и 120 синапсов, стоит 41 доллар).

Военные же отдают предпочтение изделиям на базе плат Balboa860 фирмы HNC. Плата Balboa содержит векторно-конвейерный процессор i860 (пиковая производительность 100 MFLOPS) и до 24 Мб специально организованной оперативной памяти. Для Balboa предусмотрен набор дочерних модулей для обработки изображений, подключения к высокоскоростным сетям и так далее. Платы семейства Balboa достаточно дороги — от 15 тыс. долларов, но заказчики не скупятся. Дело в том, что фирмой HNC управляет сам Хехт-Нильсен, патриарх нейроматематики и создатель первых нейронных суперкомпьютеров военного применения Mark TRW.

Банкиры тоже люди серьезные. В их играх «по-крупному» участвуют значительно более дорогие суперкомпьютеры SNAP фирмы HNC и CNAPS фирмы Adaptive Solutions. SNAP представляет собой суперкомпьютер, содержащий до 64 SIMD-процессоров SNAP-2, а также управляющую плату Balboa и специализированные модули для ввода и подготовки данных (по секрету скажем, что многие изумительные по дерзости операции финансовой группы Citicorp на мировом валютном рынке обязаны своим рождением именно компьютеру SNAP). Кстати, в престижном конкурсе Gordon-Bell Prize SNAP признан лучшим суперкомпьютером 1993 года по критерию производительность/стоимость — обогнав и Cray, и Paragon, и других признанных лидеров. Разумеется, не имеет смысла присматриваться к SNAP и ему подобным без сотни-другой тысяч долларов в кармане. Но не отчаивайтесь — у HNC есть кое-что и для клиентов поскромнее. Это:

- пакет Falcon для проверки подлинности кредитных карточек в режиме on-line;
- пакет Underwriting Colleague для оценки кредитных и инвестиционных рисков;
- пакет SkuPLAN для предсказания объемов продаж (кстати, успешно применяется как на рынке шоколада, так и при продажах прокатных станов);
- пакет AREAS для комплексной оценки недвижимости и другой собственности;

а также ряд других приложений, как массового применения, так и заказных.

Итак, нейронные сети, успешно завоевывая мир, не сегодня-завтра появятся и на нашем рынке. Тот, кто успеет заблаговременно подготовиться к их приходу — будь то коммерческий банк, компьютерный торговый дом или «вольный стрелок»-программист, получит несомненные конкурентные преимущества и оч-чень весомые дивиденды. Надо только помнить, что нейросеть — это не электрочайник, она не начнет работать без серьезной настройки, проведенной профессионалами. Тем не менее, тот, кто всерьез заинтересуется перспективами использования нейронных сетей, обязательно найдет возможность преодолеть пресловутый «барьер вхождения». ■

Некоторые из Ваших серьезных конкурентов уже используют Power Macintosh.

•

Приятных сновидений.

Они используют мощный RISC-процессор. Они работают с программами Макинтоша, DOS и Windows.*

Они больше успевают. Возможно, они больше зарабатывают. Подумайте об этом. Позвоните 978-4262 или 978-1257. Спросите телефон Вашего дилера Apple. Не теряйте время.

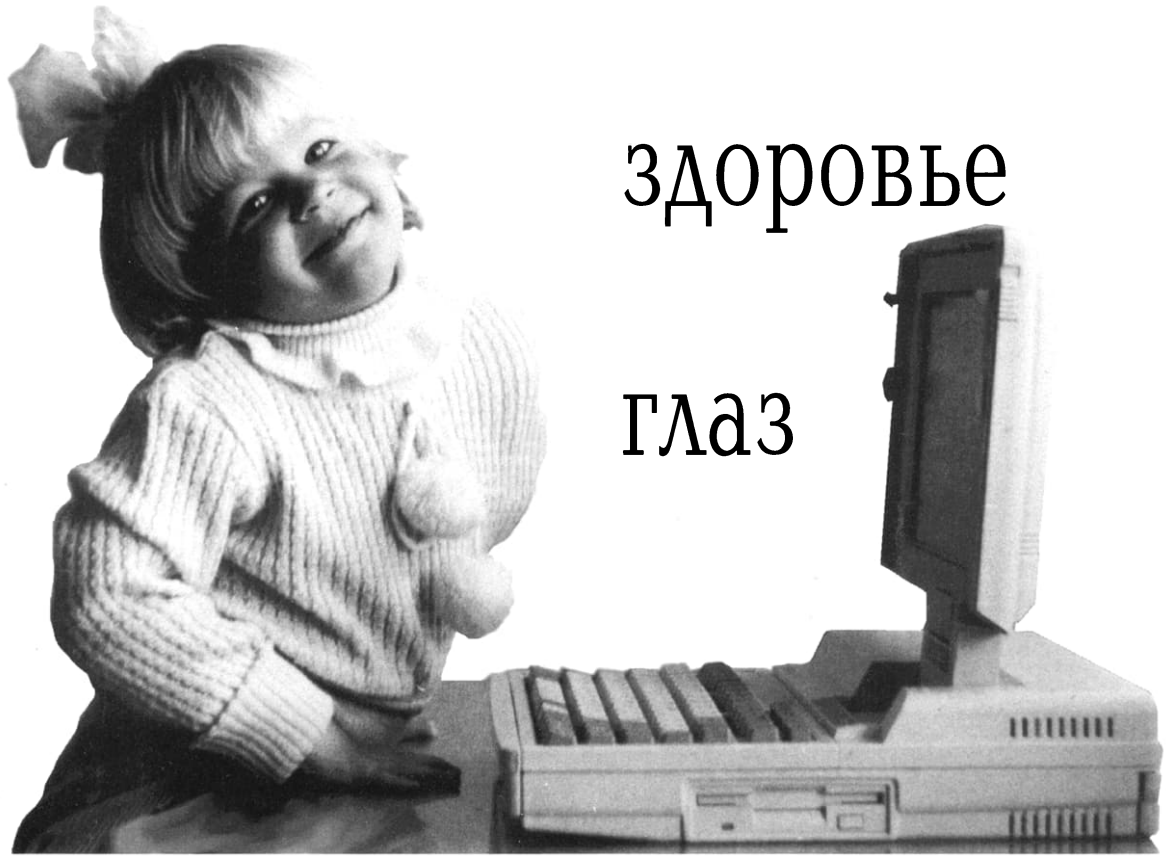
Apple

Power Macintosh уже в Москве. Чего же Вы ждете?

**Используя программное обеспечение SoftWindows™ фирмы Insignia Solutions Inc*

® RUI IMC Apple Computer Москва 103055, 2-й Вышеславцев Пер., д. 17 Тел. 978-1257/4262/2821/3562/2100, факс 978-1391. Сделано на компьютере Apple Power Macintosh.

Береги здоровье глаз



ТОО "Интерлаз" предлагает свои услуги в проведении профилактики и лечения офтальмологических заболеваний, вызываемых продолжительной и напряженной работой с компьютером или деловыми бумагами.

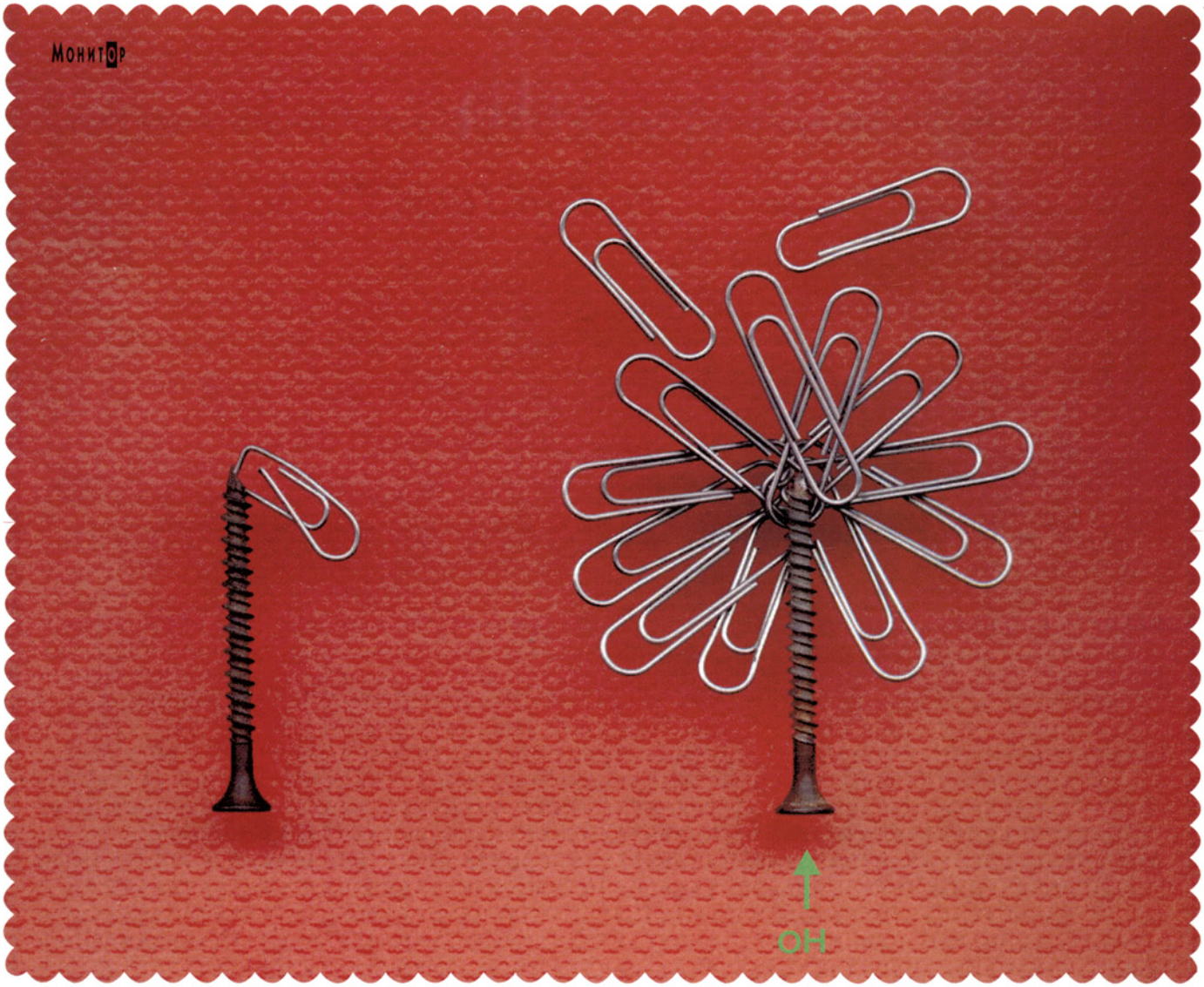
Кроме того, фирма имеет отделения кардиологии, гинекологии, пульмонологии, гастроэнтерологии, отоларингологии, неврологии, дерматологии, урологии, сексопатологии, минихирургии.

Весь комплекс лечебно-диагностических услуг проводится с применением уникальной и безопасной лазерной технологии.

Возможна оплата за лечение Ваших сотрудников по безналичному расчету.

вместе с фирмой "Интерлаз"!

ТОО фирма "Интерлаз"
Адрес: Москва, ул. Ефремова, 12а.
Тел. 246–8727, тел/факс: 246–8428



дал

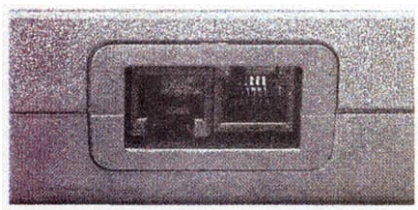
рекламу

в журнале

«МОНИТОР»

Для тех, кто понял - телефоны отдела рекламы журнала «Монитор»: 903-46-26, 238-99-11.

Корпорация LVS предлагает встроенные FAX / MODEM / Сетевые платы фирмы Ositech для блокнотных компьютеров *COMPAQ (Portable, Contura, LTE Lite)*



- ***Внутренние***, ставятся на плату расширения, не требуют навесных перемычек.

- ***Быстрые*** (16 битовая шина).

- ***Единая внутренняя карта*** для модема и локальной сети.

- ***Возможность одновременной работы*** в сети и с модемом.

- ***Поддержка основных стандартов*** (Ethernet IEEE 802.3 и Token Ring802.5)

- ***Гарантия 1 год***.

- ***Управляемое потребление энергии*** при минимальном потреблении в неактивном состоянии.

EthernatTALK - адаптер Ethernet. Совместим с большинством сетевых операционных систем. Поставляется как в конфигурации 10BaseT Ethernet, так и 10BaseT& 10Base2 Ethernet.

TokenetTALK - адаптер Token Ring. Поддерживает конфигурацию как 4, так и 16 Mbps. Поддерживает кабели STP и UTP.

LapTALK Plus 14.4K- модем данных 14.4 Kbps MNP-2-5, MNP 10, V.42. V.42 bis и факсимильной передачи Group 3 9600bps.

LapTALK Plus 2400 - модем 2400bps MNP-2-5, MNP 10, V.42. V.42 bis. и факсимильной передачи Group 3 9600bps.

CoaxTALK 3270 - адаптер coax 3270 для связи с IBM контроллером 3270 с интерфейсом типа "A".

Remote TALK 3270 - программа эмуляции взаимодействия с IBM компьютером по протоколу SNA/SDLC 3270.

Direct TALK 3270 - программа эмуляции терминала IBM 3278/79 при сохранении независимой работы процессора.

LVS CORPORATION

117342, Москва,
Профсоюзная ул.,65
Тел.: (095)330-42-66, (095)330-36-66
Факс: (095)330-36-93, (095)956-50-97

Ваш *COMPAQ* Notebook не останется один !

Вы помните, как покупали свой первый компьютер ?

Его бережно распаковывали, долго настраивали... А затем, секретарь печатал на нем свои бесконечные документы, вечером или в обед сотрудники собирались поиграть.

Мало-помалу все привыкли к компьютерному чуду. Время шло, и однажды настал тот день, когда программа, значительно облегчающая труд конкурентов, не запустилась на Вашем компьютере. С каждым разом этих программ становилось все больше, и Вы топтались на месте, когда окружающие стали уходить вперед... А потом, вдруг, компьютер просто не включился.

Что же делать? Как правильно выбрать компьютер, подходящий именно Вам, хорошего качества, с соответствующим дизайном? А самое главное, как сэкономить деньги?

ЭТО ОЧЕНЬ ПРОСТО !

Компьютеры марки CND™ обеспечены пятилетней гарантией, оснащаются любым необходимым программным обеспечением. Клиенты СКТБ Компьютерных Сетей не имеют хлопот не только с компьютерами и программными продуктами, но и с различными компьютерными аксессуарами, носителями информации, мультимедиа, периферией и локальными вычислительными сетями. Все это можно приобрести в фирменном салоне-магазине.

Консультанты помогут Вам подобрать компьютер, именно для Ваших целей. Специалисты установят программное обеспечение. Причем Вы сможете значительно сэкономить деньги, - Вы приобретете технику с гарантированным качеством, оптимальной конфигурации и с лучшим для Вас соотношением цены и производительности непосредственно у фирмы-производителя.

Приходите к нам, мы всегда рады Вас видеть!

ВОСХОЖДЕНИЕ НА КОМПЬЮТЕРНЫЙ ОЛИМП

energy
EPA POLLUTION PREVENTER

intel inside

Мы используем только настоящие микропроцессоры Intel в наших качественных компьютерных системах.

Microsoft®
SOLUTION PROVIDER

СКТБ Компьютерных Сетей
Москва Севастопольский проспект д.43 Б
тел. (095) 127-9011; факс (095) 127-7990
Email cnd@network.msk.su.
✉ 113186 Россия Москва а/я 263

All trademarks are property of their respective owners.

Компьютерный «ларёк» 1.95

Код	Продукт	Краткая характеристика	Объем поставки	Цена $	Требования к оборудованию
MS-01	Data Scope	Интегрированный пакет для многомерного статистического анализа данных и прикладной статистики.	2 дискеты + документация	300	IBM PC/AT, математический сопроцессор, 2,5 Мб на HDD, видеокарта 256 кб
MS-02	АРИСС	Пакет для предсказания будущих значений отдельных динамических рядов с использованием АРИСС-методов, включающий авторегрессии, интегрирование, скользящие средние.	1 дискета + документация	36	IBM PC/AT
SL-01	Turbo Games 3.0	Библиотека для Turbo Pascal 6.0-7.0. Спрайты, трехмерная графика, стереоэффекты. Доступ к ZIP файлам. Поставка в исходных текстах.	1 дискета + документация	15	IBM PC EGA/VGA
SL-02	Data Vision 2.0	Объектно-ориентированная библиотека для Borland Pascal 7.0. Унифицированный интерфейс с БД различных форматов (включает dBase-интерфейс, редакторы БД, QBE, генератор отчетов...).	1 дискета + документация	40	IBM PC
SL-03	DVision DBF Interface 2.0	Поддержка DBF-файлов. Работа с MEMO-полями формата dBase, Clipper и FoxPro. Поставка в исходных текстах.	1 дискета + документация	25	IBM PC
SL-04	Atoll	Библиотека растровой графики (Pascal, C). Преобразование изображений в реальном времени, видеоэффекты, зумирование, анимация.	1 дискета + документация	20	IBM PC EGA/VGA
SL-05	SPRight Lib	Объектно-ориентированная библиотека для Borland C++ 1.xx-2.xx. Спрайты, спрайтовые объекты, звуковые эффекты, поддержка графического формата PCX. Прозрачна для большинства графических библиотек. Поставка в исходных текстах.	1 дискета + документация	50	IBM PC/AT EGA/VGA
SL-06	MAESTRO	Библиотека для Turbo Pascal и C++.Воспроизведение звуковых эффектов и музыки. Поддержка PC Speaker, Covox (LPT1-LPT3), Sound Blaster (DMA и Direct Mode).Совместимость с синтезатором.	1 дискета + документация	40	IBM PC/AT
SL-07	ПараФорт	Форт-система (поддержка стандарта FORTH-83). Турбосреда. Оконная библиотека. Встроенный форт-ассемблер.	1 дискета + документация	30	IBM PC/AT
SL-08	Форт-Ягуар	32-разрядная форт-система для защищенного режима.Полное использование всей оперативной памяти компьютера.Встроенный форт-ассемблер, вещественная арифметика, турбосреда, оконная библиотека.	1 дискета + документация	70	IBM PC 386
SL-09	Super Vision 3.0	Интегрированный графический пакет для Borland Pascal 7.0, MS-DOS. Визуальное проектирование интерфейса. Мощная иерархия стандартных объектов. Поддержка EGA, VGA, SVGA; DPMI. Работа с форматами GIF, TIFF, PCX. Сохранение идеологии Turbo Vision. Исходные тексты более 1,7 Мб.	3 дискеты + документация	119	IBM PC/AT EGA/VGA/SVGA «мышь»

Код	Продукт	Краткая характеристика	Объем поставки	Цена, $	Требования к оборудованию
SC-01	BIC	Графический редактор для VGA (640x480x16). Основной формат – PCX. Экспорт-импорт файлов (Borland Pascal & C). Описание фрагментов константами на языках Pascal и C. Гипертекстовая контекстная помощь.	1 дискета	20	IBM PC 1,2 Мб на HDD 512 кб ОЗУ
SC-02	TBG	Библиотека для программирования графических приложений (BGI). Создание логической структуры изображения. Манипуляция изображениями сложной топологии. Графический редактор.	1 дискета	20	IBM PC/AT EGA/VGA «мышь»
SC-03	Турбограф	Графический редактор с генерацией кода для компиляторов Borland. Визуальное программирование на Turbo Pascal, TurboC, Turbo C++. Работа во всех графических режимах, поддерживаемых библиотекой фирмы Borland.	1 дискета	20	IBM PC
SU-01	FDA v. 6.0	Анализ и точное копирование гибких магнитных дисков всех типов. Самостоятельное создание ключевых меток любого уровня сложности. Быстрое и качественное форматирование всех видов дискет любого формата, в том числе нестандартного.	1 дискета	35	IBM PC/AT
SU-02	ALL.BGI	Набор русифицированных графических шрифтов в формате BGI. Набор драйверов в стандарте BGI: VGA 320x200x256, SVGA 640x400x256, 640x480x256,, 800x600x256, 1024x768x256.	3 дискеты	10	IBM PC/AT
SU-03	FDREC	Программа восстановления поврежденных файлов на дискетах всех типов.	1 дискета	20	IBM PC/AT

Оплата принимается только в рублях по текущему курсу доллара.
Адрес для писем: 117419 Москва, а/я 765.
Тел.: (095) 943-99-11. Факс: (095) 238-99-11.

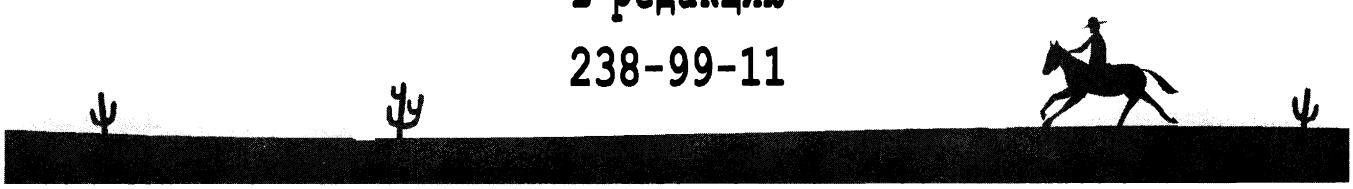
НОВОСТЬ!

ПРОКАТ ИГР НА CD-ROM

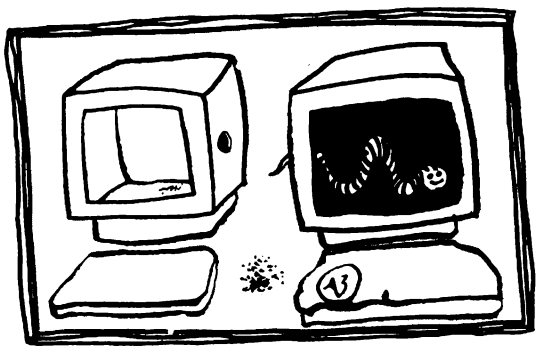
По всем вопросам обращаться

в редакцию

238-99-11



Применение программного обеспечения удаленного доступа The Norton pcANYWHERE



PC

А. Замятин

В настоящей статье рассматривается семейство программных продуктов The Norton pcANYWHERE 4.5, которое включает в себя три отдельных пакета: Host, Remote и LAN (в дальнейшем по тексту AWHOST, AWREMOTE и AWLAN соответственно, общее наименование AW). Все они выпущены фирмой «Symantec Corp.» в 1991 году.

Программные продукты AW предназначены для организации удаленного доступа к ресурсам компьютера с использованием другого компьютера или терминала [1], [2].

Соединение может осуществляться при помощи стандартных коммуникационных средств (последовательные порты RS-232C) через модем или напрямую либо с использованием сетевых средств. В последнем случае поддерживаются протоколы NetWare/IPX и NetBIOS.

Скорость обмена информацией через последовательный порт может достигать 115200 бит/с; при использовании сети связь устанавливается с максимально доступной для сети скоростью.

Для использования (запуска) программного обеспечения AW необходимы следующие аппаратные и программные средства: IBM PC/XT, AT, PS/2 или совместимая с ними рабочая станция, работающая под управлением MS-DOS 3.1 и выше; не менее 320 кб ОЗУ, доступного для собственно AW. Пакет AWLAN, как уже отмечалось, поддерживает сетевой протокол Novell NetWare/IPX, а также NetBIOS, стандартный сетевой протокол, разработанный фирмой IBM. Большинство сетевых операционных систем (включая NetWare) имеют встроенную поддержку NetBIOS.

При работе с AW различают три основных вида компьютеров (по типу выполняемой операции): Host, Remote и Gateway. Возможный перевод этих терминов на русский язык — головной, удаленный и шлюз; в дальнейшем будут использованы только оригинальные наименования. Остановимся подробнее на каждом из них.

Host-компьютер — это компьютер, позволяющий осуществлять доступ к своим ресурсам с использованием другого компьютера. Причин для подобного доступа может быть множество. Возможна ситуация, когда пользователь, работающий за Host-компьютером, нуждается в технической поддержке или обучении. После окончания рабочего дня Host-компьютером может быть компьютер, расположенный в офисе. Или наоборот, он может быть домашней машиной, в то время как хозяин находится на работе или в пути. Во всех ситуациях Host-компьютер является «ведомым» (то есть подчиняется указаниям «ведущего»).

Программные средства AW позволяют осуществлять полный удаленный контроль работы Host-компьютера: при этом на экране Remote-компьютера отображается такая же информация, что и на экране Host-компьютера; ввод информации с клавиатуры Remote-компьютера Host-компьютер воспринимает так же, как и ввод информации со своей клавиатуры; Host-компьютеру доступна «мышь» Remote-компьютера). Предусмотрены развития средства для организации двунаправленного высокоскоростного обмена файлами в фоновом или обычном режиме. При осуществлении удаленного контроля предоставляется также возможность блокирования экрана и клавиатуры Host-компьютера. При графической работе видеоадаптера поддерживаются все доступные режимы; в случае необходимости осуществляется автоматическая трансляция графики.

Пользователь Remote-компьютера является «ведущим» в процессе удаленного доступа. Он может контролировать процесс выполнения

задачи на Host-машине, самостоятельно запускать на ней задания на выполнение, пересылать файлы с Host-машины на Remote-машину или наоборот. Также возможно вмешательство (содействие) в работу пользователя Host-компьютера с целью разрешения возникшей у него проблемы.

Работа Host и Remote-компьютеров проиллюстрирована на рис. 1 (телекоммуникационный сеанс) и рис. 2 (сеанс связи внутри локальной сети).

При работе в локальной сети обычно одна из сетевых станций при помощи модема подключается к телефонной линии. Однако возникают ситуации, когда любому из пользователей сети может потребоваться телекоммуникационный доступ. Разрешить возникшую проблему позволяет Gateway-операция. Использование Gateway-компьютера дает возможность пользователю любой рабочей станции сети осуществить выход на Gateway-модем для организации сеанса связи.

Вариант организации соединения с использованием Gateway-компьютера представлен на рис. 3.

Более сложный вариант взаимодействия с использованием двух Gateway-компьютеров изображен на рис. 4.

Процесс инсталляции всех пакетов AW очень прост и максимально автоматизирован. Для начала установки необходимо запустить программу INSTALL.EXE (она находится на первой дискете дистрибутивного комплекта AWHOST, AWREMOTE и AWLAN). Весь диалог осуществляется при помощи

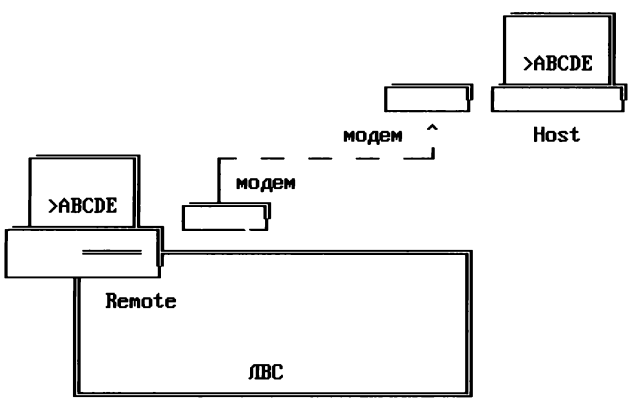


Рис. 1. Взаимодействие Host и Remote через модем

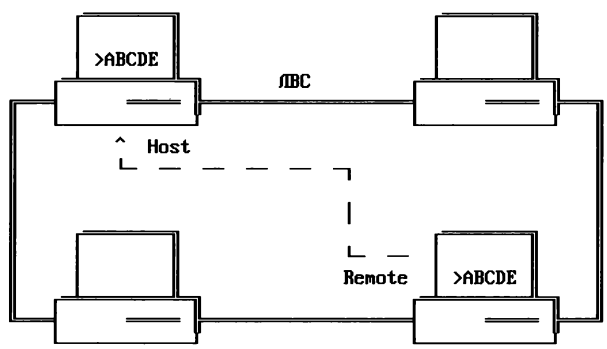


Рис. 2. Взаимодействие Host и Remote через сеть

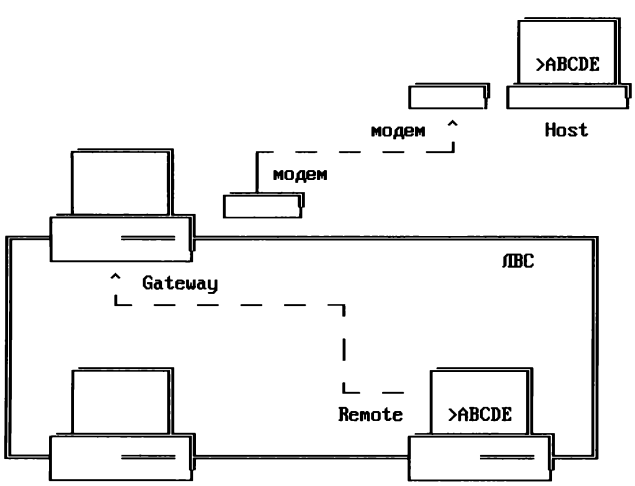


Рис. 3. Взаимодействие Host и Remote через Gateway

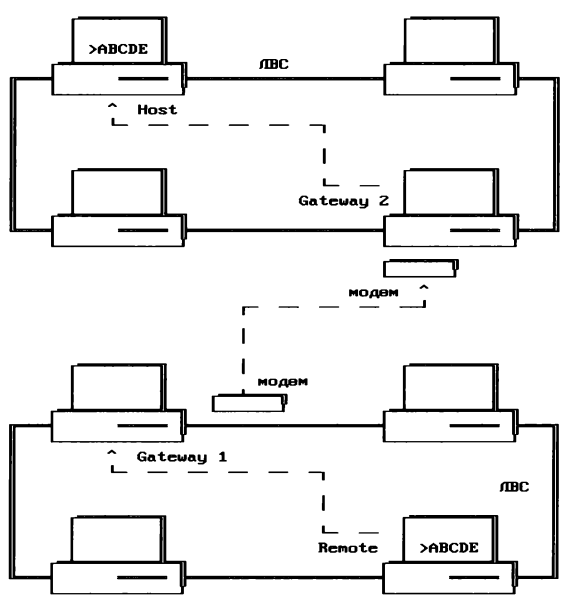


Рис. 4. Взаимодействие Host и Remote через два Gateway

меню. Справочная информация доступна после нажатия <F1>. Единственная записываемая пользователем в соответствующих полях ввода информация — полный путь для размещения файлов AW и наименование организации — владельца программного обеспечения. Рекомендуемые каталоги для AW — \AWHOST, \AWREMOTE и \AWLAN.

По окончании процесса установки AW необходимо произвести настройку его конфигурации в соответствии с используемыми аппаратными средствами. Рассмотрим подробнее конфигурирование AWREMOTE и AWHOST для случая прямого соединения через последовательный асинхронный интерфейс [3].

После запуска программы AWREMOTE.EXE в ее главном меню выбирается пункт «Hardware Configurations...» и добавляется (нажатием <INS>) новая конфигурация — прямое соединение. Рекомендуемое наименование — «Serial: Direct». Возможные параметры конфигурации показаны на рис. 5.

Далее после выбора пункта главного меню «View/Modify List Of Hosts...» производится ввод условного наименования нового Host-компьютера (например, «ANYWHERE direct host») и заполняется информация о нем (рис. 6).

Для AWHOST.EXE также задается конфигурация прямого соединения (с помощью пункта «Hardware Configuration...» главного меню), после чего эта конфигурация объявляется активной (пункт «Select Active Configuration»).

На этом первоначальная настройка программного обеспечения удаленного доступа может быть закончена. С другими изменяемыми параметрами можно познакомиться непосредственно в процессе работы с AW.

Для начала сеанса связи с использованием прямого соединения последовательных портов необходимо на одном компьютере загрузить программу AWHOST.EXE, на другом — AWREMOTE.EXE.

В главном меню AWHOST.EXE требуется выбрать пункт «Begin Host Operation...» и далее — «Wait For A Connection». В меню AWREMOTE.EXE активизируется пункт «Connect Via List Of Hosts...», после выбора соответствующего условного наименования Host-компьютера (в нашем случае — «ANYWHERE direct host») сеанс связи начинается нажатием <F2>.

Другой вариант инициализации соединения заключается в использовании пункта «Quick Connect...» главного меню REMOTE.EXE и его подпункта «Direct Connect/Terminal Mode» (рис. 7). В этом случае может потребоваться изменение параметров соединения при помощи «Modify Connect Parameter».

После начала сеанса связи (установления соединения) пользователь Remote-компьютера получает полный удаленный доступ ко всем ресурсам Host-компьютера; экран

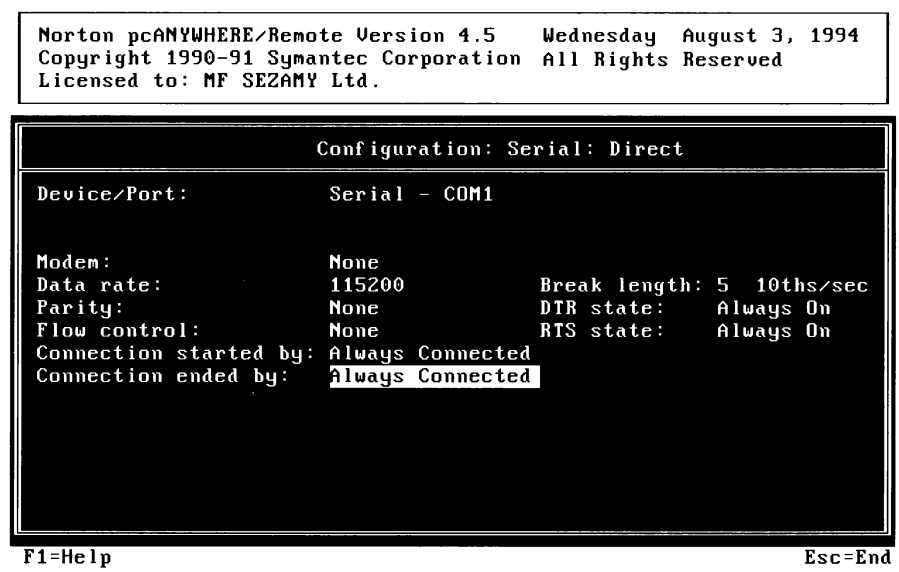


Рис. 5. Шаблон для ввода параметров конфигурации AWREMOTE

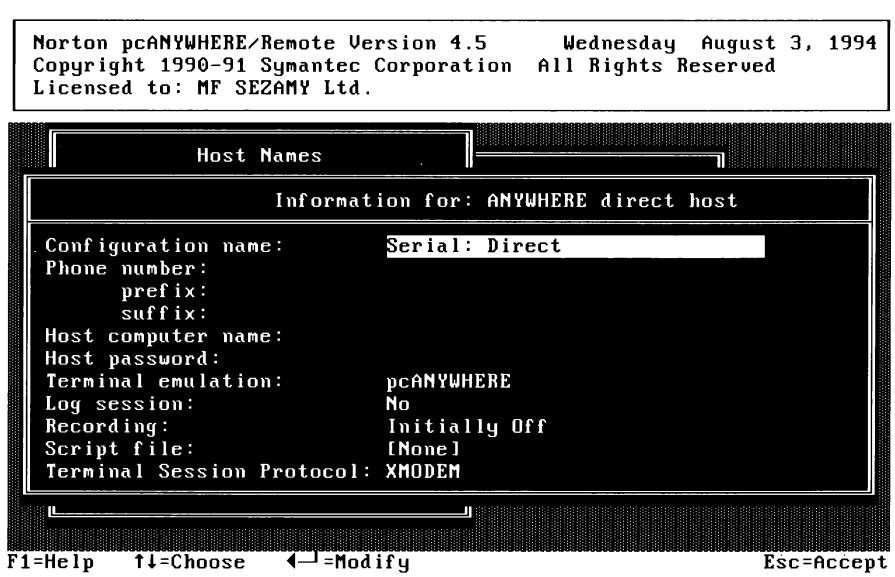


Рис. 6. Шаблон для ввода информации о Host

Remote являет собой копию Host-экрана.

В случае, если возникает необходимость пересылки файлов с Remote на Host-компьютер или наоборот, можно использовать специальную программу AWSEND.EXE, которая располагается в каталоге AWHOST. После запуска AWSEND без параметров командной строки на дисплее появляется полноэкранный интерфейс, показывающий каталоги и файлы Host и Remote-компьютера одновременно. Выделение (выбор) необходимых файлов осуществляется нажатием <Пробел>. Действие над выделенными файлами производится с помощью «горячих клавиш» (обозначены в нижней строке экрана) или системы меню (рис. 8).

Во время сеанса связи с помощью специальной комбинации клавиш (по умолчанию — <Ctrl><R-Shift>) Remote-пользователем может быть осуществлен доступ к системе меню «Session Options» (рис. 9). Выбором пункта «End Session...» соединение прерывается. В случае необходимости обмена сообщениями можно использовать режим «Chat With Host User». При этом на экране «всплывают» два окна, в которых отображается вводимая с клавиатуры Host и Remote-компьютеров символьная информация.

Программа AWLAN.EXE включает в себя возможности AWHOST и AWREMOTE и, кроме того, содержит много дополнительных опций. К основным из них относятся доступ к сетевым конфигурациям и Gateway-операции.

Для использования компьютера в качестве Gateway (рис. 3, 4) необходимо запустить на нем программу AWLAN.EXE и выбрать в главном меню AWLAN пункт «Gateway Operation...» (рис. 10). После этого требуется сконфигурировать Gateway; для этого при помощи подпунктов «Select Incoming Configuration...» и «Select Outgoing Configuration...» задаются параметры соединения с Remote и Host-компьютером соответственно (рис. 11). Если применяются два Gateway-компьютера, то за «incoming» (входящее) принимается соединение, идущее по направлению от Remote, а за «outgoing» (выходящее) — от Host-компьютера.

После определения параметров конфигурации выбирается подпункт «Activate The Gateway...», и Gateway-компьютер начинает выполнять свои функции в фоновом режиме. Для отключения (выгружения из памяти) программного обеспечения Gateway, необходимо повторно запустить AWLAN.EXE и в меню «Gateway Operation...» выбрать подпункт «Exit And Remove From Memory...».

Программы AWHOST.EXE, AWREMOTE.EXE, AWLAN.EXE и AWSEND.EXE предоставляют возможность использования командной строки для определения

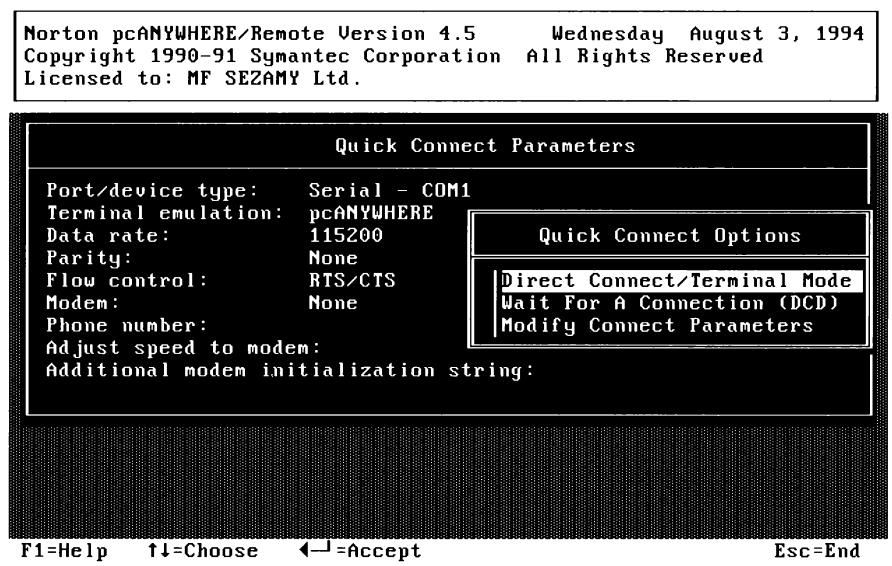


Рис. 7. Система меню AWREMOTE «Quick Connect...»

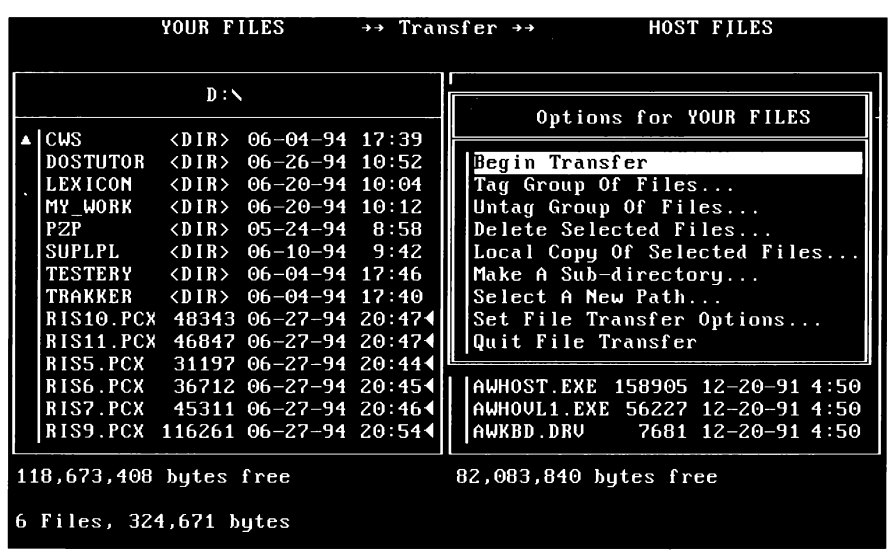


Рис. 8. Интерфейс программы AWSEND

режимов работы. Полноэкранный интерфейс при этом не задействуется. Запуск указанных программ с использованием аргументов командной строки может быть особенно полезен при использовании в команд-

Рис. 9. Система меню «Session Options»

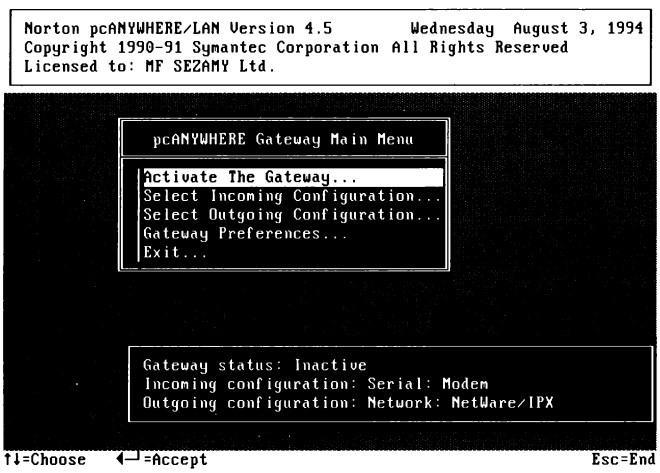


Рис. 10. Система меню AWLAN «Gateway Main Menu»

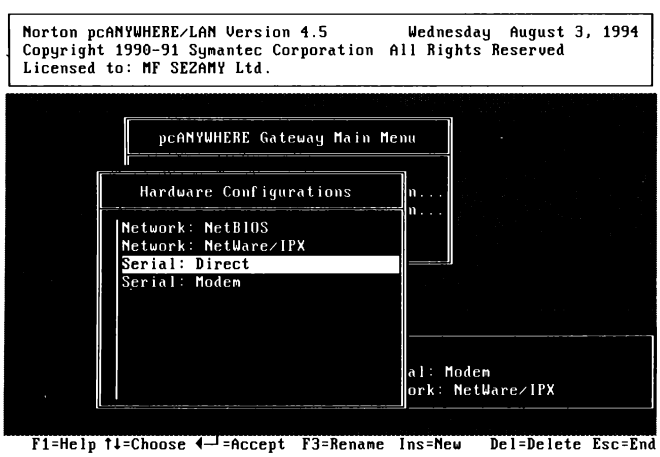


Рис. 11. Определение Gateway-конфигурации

ных файлах и формировании с использованием AW автоматизированных систем.

Еще одним важным средством построения распределенных автоматизированных систем является развитый язык сценария, поддерживаемый программным обеспечением AW.

Исходный текст сценария набирается в формате ASCII в любом текстовом редакторе. Файл с исходным текстом программы может иметь произвольное имя и обязательное расширение SCR.

Трансляция файла .SCR осуществляется в режиме «Utilities — Preprocess A Script ...». При этом создается файл с тем же именем и расширением SCX. В случае возникновения ошибок последние находят свое отражение в одноименном файле с расширением ERR.

Откомпилированный файл сценария может быть выполнен одним из следующихспособов:

- автоматически после осуществления успешного соединения с Host-компьютером. Для этого в поле «Script file» в таблице информации о Hoste-компьютере (рис. 6) должно быть занесено имя выполняемого сценария;
- при осуществлении сеанса удаленного доступа активизацией меню «Session Options» (рис. 9) и дальнейшим выбором пункта «Run Script...»;
- указанием имени файла сценария в соответствующем аргументе командной строки.

В заключение следует отметить, что на российском рынке появилась новая улучшенная и дополненная версия рассматриваемого программного обеспечения — The Norton pcANYWHERE 5.0 for DOS. Согласно информационным материалам [4], в числе новых свойств пакета фирма Symantec отмечает:

- 20-процентное повышение скорости передачи данных;
- улучшенное управление памятью;
- повышение безопасности данных;
- упрощенное управление удаленным доступом в сетях. ■

Литература

1. **The Norton pcANYWHERE. User's Guide.– USA: Symantec Corporation, 1991.**
2. **The Norton pcANYWHERE/LAN. User's Guide.– USA: Symantec Corporation, 1991.**
3. **Zamyatin A.J., Dobrodzej J., Mazurkiewicz A., Piekoszewski W., Szczerek M. Installation, setting and usage of The Norton pcANYWHERE – remote-access protection for multi-station semi-automatical triboresearching systems: Laboratory-scientific research («Computer-aided Tribology» series).– Radom: MCNEMT, 1994.**
4. **К. Ахметов, А. Борзенко, Д. Рогаткин. Новости: Symantec – новые продукты и новые планы// КомпьютерПресс.– 1994.– № 6.**

ARS MOSCOW

HEWLETT PACKARD
Authorized Wholesaler

НАДЕЖНОСТЬ, ПРОВЕРЕННАЯ ВРЕМЕНЕМ

Немедленная поставка со склада в Москве

Крупнейший склад оборудования HEWLETT-PACKARD в России

HP Server PCs

Мощные серверы и рабочие станции высочайшей надежности, сетевые продукты

HP Peripherals

Лучшие в мире лазерные и новейшие струйные принтеры.
Многоцветные графопостроители и сканеры

HP Accessories

Расходные материалы для HP оборудования

Приглашаем для совместной работы серьезных региональных и отраслевых партнеров с предоставлением статуса
"Authorized Dealer HEWLETT-PACKARD"

☆ **Лучшие цены в России**

☆ **Специальная дилерская программа**

119-8824, 316-7627, 110-5830
119-0909, 316-8328, 230-6808

Fax: 119-6841

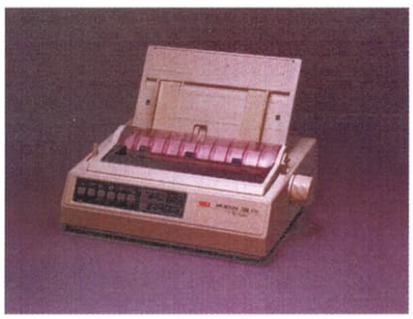
Адрес: 113149, Москва, ул. Сивашская, д.4, кор. 2

Гарантия HEWLETT PACKARD до 3-х лет

Nexus

Фирма НЕКСУС
Москва, Кузнецкий мост, д. 21/5
921-36-70, 928-23-67

OKI
People to People Technology



MICROLINE 321 Elite

Таблица сравнительных характеристик

	ML 320/321	EPSON FX 1050
Скорость печати	360 зн/сек	290 зн/сек
Плотность зн/дюйм	10,12,15,17,20	10,12,17,20
Число копий	Ориг+3	Ориг+3
Буфер	15 кб	8 кб
Печать графики	в две стороны	в одну сторону
Надежность головки	200 млн.симв	100 млн.симв.

MICROLINE 521 elite

Таблица сравнительных характеристик

	ML 520/521	EPSON FX 1170
Скорость печати	433 зн/сек	380 зн/сек
Плотность зн/дюйм	10,12,15,17,20	10,12,17,20
Число копий	Ориг+5	Ориг+3
Буфер	64 кб	24 кб
Печать штрих-кодов	да	нет
Надежность гловки	200 млн.сив.	100 млн.симв.

OL 400e, OL 410e

Таблица сравнительных характеристик

	OL 400e	OL 410e	HP 4L
Метод печати	LED	LED	LASER
Разрешение dpi	300	300 (600 RET)	300 (600 RET)
Память (норм/макс.)	512kb/4.5 Mb	1/5 Mb	1/2 Mb
Ресурс барабана	20 000 копий	20 000 копий	3000 копий
RS 232	есть	есть	нет
Команды	PCL 4	PCL 5	PCL 5

Nexus

Фирма НЕКСУС
Москва, Кузнецкий мост, д. 21/5
921-36-70, 928-23-67

OKI
People to People Technology

MICROLINE 390FB

Таблица характеристик	
Скорость печати	270 зн/сек
Плотность зн/дюйм	10,12,15,17,20
Число копий	Ориг+4
Буфер	20 кб
Совместимость	EPSON LQ
Надежность головки	200 млн.симв.
Интерфейс RS 232	дополнительно

MICROLINE 3410

Таблица сравнительных характеристик		
	ML 3410	**EPSON DFX 5000**
Скорость печати	550 зн/сек	533 зн/сек
Плотность зн/дюйм	10,12,15,17,20	10,12
Число копий	Ориг+7	Ориг+5
Буфер	16кб	3 кб
Автоподача листов	дополнительно	нет
Наработка на отказ	8000 час.	6000 час.

Цены (розничные/дилерские)

ML 320 (узкий).....380/350$
ML 321 (широкий).....440/400$
ML 520 (узкий).....460/420$
ML 521 (широкий).....510/460$
ML 590 (узкий).....590/530$
ML 591 (широкий).....640/580$

ML 390FB.....1100/1000$
ML 3410.....1250/1150$
OL 400e..... 650/600$
OL 410e.....750/680$
OKIFAX OF 1000.....1900/1750$
Факс на обычной бумаге

Приглашаются дилеры.

Звуки мультимедиа в среде MS-DOS

PC

В. Дьяконов

Итак, вы решились и приобрели средства мультимедиа! Поздравляю. Вы, конечно, отважный человек, пожертвовавший своим материальным достатком ради освоения новых достижений информатики. Впрочем, они дадут вам не только удовольствие (говорят, и за это надо платить), но и бесценные знания и, возможно, даже новые заработки.

Автор в вводной статье [1] достаточно популярно описал, как устанавливаются аппаратные средства мультимедиа и как инсталлируются обслуживающие их драйверы. Материал статьи интересен тем, кто имеет хотя бы начальные знания в области грамотного применения программных продуктов мультимедиа под MS-DOS. Если вы не знаете, что такое файлы autoexec.bat и config.sys, можете на этом закончить чтение статьи. Вам (не обижайтесь) надо пройти ликбез по компьютерам и их программированию. Если же вы хоть немного спец — с Богом дальше!

Материал этой статьи посвящен описанию только легально полученных программ для работы в MS-DOS. Это модули, входящие в программное обеспечение CD-ROM GCD-R400B и платы Audio Excel Pro 16, а также специально выбранные программы из приобретенного автором лазерного диска «14000 Programs». Эти продукты относятся к категории ShareWare, то есть могут свободно описываться и распространяться.

Итак, после инсталляции драйверов CD-ROM и звуковой платы вы обнаружите, что конфигурационный файл config.sys и загрузочный пусковой файл вашего ПК подверглись некоторой модификации. Автор не любитель захламлять эти файлы и поэтому приводит их лишь с небольшими комментариями. Листинг 1 показывает, как выглядит файл config.sys принадлежащего автору ПК 386DX после инсталляции драйверов и косметической обработки:

Листинг 1

```
DEVICE=C:\DOS\SETVER.EXE
rem ****** Отмена ограничения на имя накопителя ********
LASTDRIVE=Z
rem ***********************************************
BREAK=ON
DEVICE=C:\DOS\HIMEM.SYS
DOS=HIGH,UMB
BUFFERS=1
FILES=30
DEVICE=C:\DOS\EMM386.EXE  ram
SHELL=C:\DOS\COMMAND.COM  C:\DOS\   /p
rem ******* Установка драйвера CD-ROM ****************
DEVICEHIGH=C:\CDROM\GCDR400.SYS /D:MSCD000 /B:320 /M:D /T:3
rem ***********************************************
rem  STACKS=9,256
```

Звездочками выделена единственная строка, которая появилась в этом файле в результате инсталляции драйвера CD-ROM фирмы GoldStar типа GCDR400.SYS. Если вы применяете иной CD-ROM, то он, естественно, будет задаваться иным названием. Однако до этого надо позаботиться об отмене ограничения на имена накопителей. Можно сразу ввести строку вида LASTDRIVE=Z, настроив систему на любое (в пределах от A до Z) число возможно используемых накопителей.

После указания драйвера (слово DEVICEHIGH указывает на его загрузку в верхнюю область памяти), вместе с путем доступа к нему, со знаками «/» указан ряд параметров-ключей. Ключ «/D:Имя_драйвера» дает связь с пусковым исполняемым файлом драйвера, ключ «/B:Адрес» задает адрес ввода-вывода в виде шестнадцатиричного числа, ключ «/M:Мода» задает моду (в нашем случае мода D означает DMA), и наконец ключ «/T:Число» задает порядковый номер DMA.

С оставшимся содержимым этого файла ознакомьтесь сами по описанию MS-DOS 6.0 или 6.2. Отметим лишь, что в файле задана загрузка MS-DOS в расширенную область памяти и использование драйвера emm386 этой памяти. В вашем распоряжении остается 630 килобайтов основной памяти.

А что же в файле autoexec.bat? Теперь он выглядит, как показано в листинге 2.

Листинг 2

```
@ECHO OFF
PROMPT $p$g
PATH C:\UT;C:\WINDOWS;C:\DOS;C:\NC;C:\ARC;
C:\DOS\SMARTDRV.EXE 1028 512>nul
SET TEMP=C:\DOS
LH MMOUSE>nul
LH PU_1700>nul
LH gamma>nul
LH unikbd /IB5>nul
rem ******* Установка MS-DOS драйвер CD-ROM********
LH mscdex.exe /D:MSCD000 /M:15 /E
rem ******* Запуск микшера платы звука **************
SET MC=C:\MCPRO
C:\MCPRO\MCPSVOL /M:11 /V:9 /L:11 /F:11 /C:11 /X:3
rem ***********************************************
LH C:\NC\NC
```

Здесь имеется также выделенное звездочками обращение к специальной программе работы с CD-ROM mscdex.exe. Эта программа (ее можно рассматривать как универсальный драйвер для CD-ROM различных типов) имеется в старших версиях MS-DOS 6.0/6.2 и поставляется в комплекте с программным обеспечением накопителя. Ключ «/D:Имя» связывает эту программу с установленным «персональным» драйвером CD-ROM. Имя типа MSCD000, MSCD001 и так далее задается по рекомендации фирмы Microsoft. Ключ «/M:Число» задает число блоков памяти для кеширования CD-ROM, а ключ «/E» частично загружает драйвер в верхнюю область памяти. Команда LH (сокращение от LOADHIGH) позволяет разместить драйвер целиком в верхней области памяти.

Следующие две рабочие строки относятся к управлению платой звука. Первая строка указывает путь к драйверам и программам заданной платы Audio Excel Pro 16. Вторая обеспечивает загрузку специальной программы, задающей начальные уровни сигналов: общего от всех источников — M, синтезируемых голосов — V, линейного входа — L, синтезируемой музыки — F, проигрывателя аудиодисков — C и микрофона — X. Соответствующие числа задают уровни сигналов. Эту программу нет нужды располагать в верхней области памяти — задав нужные установки на плате звука, она покидает ОЗУ автоматически. Разумеется, для других плат содержимое этих строк может быть иным. Но важен принцип!

Здесь полезно обратить внимание на отсутствие установок адреса ввода/вывода, DMA и номера прерываний платы — параметров, назначение которых может потребовать много времени. Эти параметры, как описывалось в статье [1], устанавливаются с помощью перемычек на самой плате и никоим образом не должны совпадать с установками CD-ROM и других узлов ПК. Рекомендуется также отключить игровой порт на плате звука, если он уже есть на какой-либо другой плате вашего ПК (например, на мультиплате). Все это поможет вам избежать нередких конфликтов между драйверами (SoftWare) и установками на платах (HardWare).

Тестирование мультимедиа в MS-DOS

Как правило, CD-ROM и плата звука поставляются с программами тестирования, работающими в среде MS-DOS. Впрочем, первая проверка CD-ROM может осуществляться пуском его в режиме проигрывания аудиодисков, а также путем наблюдения за его работой как нового накопителя.

Более сложным является тестирование платы звука. Но и оно резко упрощается, если есть программа тестирования. Для нашей конкретной платы такая программа называется MCPTEST.EXE. Она определяет соответствие параметров платы возможным их значениям и проверяет плату на синтез голосов, музыки и запись/воспроизведение с микрофона. На рис. 1 показано переднее панно этой программы после проведения тестирования.

Программа проверяет установки адреса ввода-вывода, прерывания и DMA звуковой платы. Если обнаружен конфликт с установками в других платах ПК, то программа сообщает от этом и прекращает

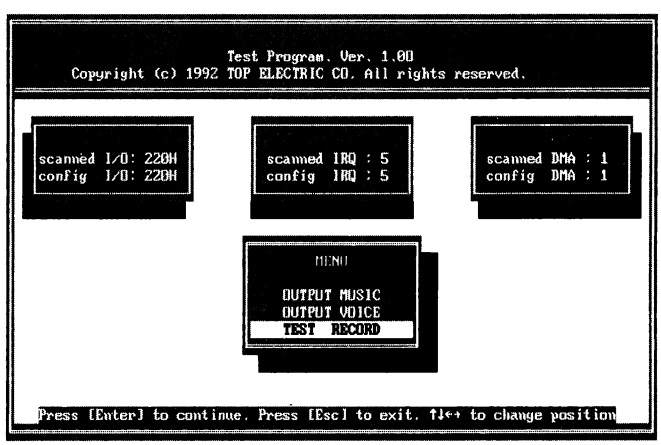


Рис. 1. Переднее панно программы тестирования звуковой платы

дальнейшую работу. В этом случае необходима перестройка параметров HardWare звуковой платы. Если нужно только выяснить текущие установки параметров звуковой платы, можно использовать более простую программу MPCSENV.

Регулировка громкости и микширование звуков

Установка номинальных уровней сигналов от всех источников звуковой платы с помощью программы MCPSOLV.EXE, пускаемой в файле autoexec.bat, уже описывалась. Но для платы Audio Excel есть еще одна более «продвинутая» программа для смешивания с разным уровнем (микширования) звуков от различных источников — MCP-MIX.EXE. Это резидентная программа, которая после пуска постоянно находится в памяти вашего ПК, и ее можно вызвать одновременным нажатием обеих клавиш Alt (правой и левой). В результате появится табло микшера (рис. 2), управление которым настолько очевидно, что не стоит подробно его описывать.

АППАРАТУРА

Многие средства мультимедиа, в частности поддержка CD-ROM и платы звука, реализованы программами под MS-DOS.

Вы можете попробовать запустить свой CD-ROM на проигрывание звукового диска в фоновом режиме с помощью программы CDPLAYER.EXE и произнести на фоне музыки монолог. Это полезно для точного подбора уровней сигналов, ибо если усиление сигнала от микрофона слишком велико, то наблюдается акустическое самовозбуждение системы, при котором колонки истошно вопят. Этот эксперимент подготовит вас к более сложному эксперименту с мультимедиа — записи своего голоса на фоне музыки и любых синтезируемых звуков. От этого рукой подать до основных применений мультимедиа — презентации разработок, услуг или товаров, подготовки учебных программ и так далее.

Воспроизведение и запись звуковых файлов

Как отмечалось, звуковые сигналы в системе мультимедиа подвергаются цифровой обработке (например, квантованию, компрессии-декомпрессии и фильтрации). Естественно, что они не имеют ничего общего с сигналами, используемыми в обычных магнитофонах и прямо-таки поедающими мегабайты памяти на магнитной ленте или диске. Существует множество форматов звуковых файлов с разными частотами квантования и различным характером цифрового представления. Отметим наиболее характерные:

.VOC — файлы с синтезируемыми голосами,
.WAV — файлы с записью оцифрованных звуковых сигналов,
.ROL — файлы музыкальных роликов (например, рок'н'ролла),
.MID — файлы с многоканальной записью по стандарту MIDI,
.NOT — файлы музыкальных произведений, исполняемых по нотам,
.CMF — файлы электронной музыки, создаваемой с помощью встроенного синтезатора, причем указания на музыкальные инструменты содержатся в самих файлах.

Платы звука с хорошим программным обеспечением обычно имеют 3-4 программы для проигрывания части из указанных файлов и записи музыкальных произведений или звуков. Так, плата Audio Excel поставляется со следующими программами проигрывания звуковых файлов:

MCPPLAYV.EXE — проигрывание файлов с расширением VOC,
MCPPLAYW.EXE — проигрывание файлов с расширением WAV,
PLAYMIDI.EXE — проигрывание файлов с расширением MID.

Первые две программы используются примитивно просто — надо задать их имя в командой строке и после этого указать имя файла. Проигрывающая про-

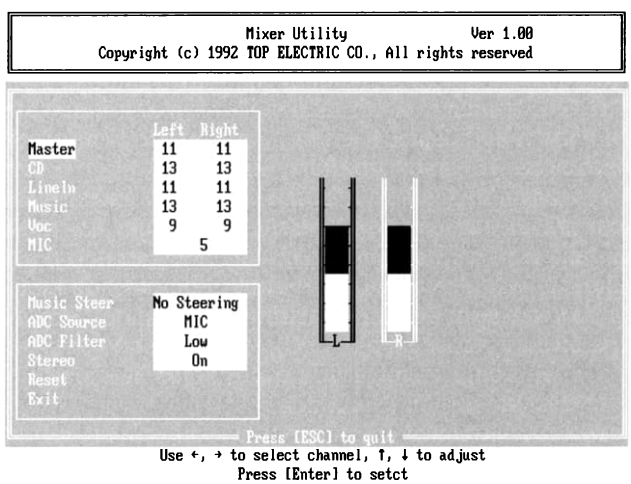


Рис. 2. Переднее панно резидентного MS-DOS микшера

грамма воспроизведет соответствующие звуки. Никаких сообщений в процессе такого рода проигрывания программы не выводят.

Разумеется, такой «сервис» не всех устраивает. Но, данная категория программ предназначена для иного — по существу, эти программы можно рассматривать как набор драйверов, включаемых в программы пользователя, подготовленные на различных языках программирования — от QBasic до C++. Сервис, сопровождающий прослушивание, пользователь создает уже сам. Мы еще вернемся к демонстрации такой возможности, а пока рассмотрим более совершенные средства для проигрывания аудиофайлов.

Одна из программ Sound Traker 2.0+ обеспечивает вывод полного перечня музыкальных инструментов исполняемого файла с расширением MOD (рис. 3). Этот перечень хранится в спецификации MOD-файлов. Для остановки проигрывания нужно нажать клавишу Esc. Если это не сделать, то, закончив исполнение вещи, программа вернется к ее началу, то есть будет бесконечно прокручивать одну и ту же вещь.

Гораздо большие удобства дает программа ModPlay Pro, которая исполняет файлы для ПК Amiga (эти ПК издавна славились великолепным синтезом музыки) с расширением MOD и NST на ПК класса IBM PC с платой звука. Программа имеет окно выбора файлов, с помощью которого легко найти нужные файлы, даже если они разбросаны по разным директориям и даже дискам (рис. 4). Она позволяет проигрывать последовательно ряд выбранных файлов. Такие программы часто называются JukeBox (проигрыватель-автомат).

Как и предыдущая программа, она выводит список имен модулей музыкальных произведений и список инструментов, отмечает звучащие в данный момент произведения нотным знаком и даже позволяет регулировать уровни звуков от проигрывателя, синтезатора, линии и микрофона — то есть она выполняет и функции микшера. Последнее очень удобно, поскольку позволяет отказаться от резидентной программы-микшера, иногда конфликтующей с другими резидентными программами.

Некоторые программы-проигрыватели довольно универсальны. Так, программа Music File/Analyzer позволяет исполнять музыкальные произведения, файлы которых записаны в форматах: MOD, ROL, VOC и других. Эта программа имеет достаточно удобный пользовательский интерфейс. При ее пуске появляется окно с главным меню (рис. 5).

Главное меню позволяет выбрать файлы из каталога, пустить на исполнение, произвести переконфигурацию драйверов системы, выполнить запись в файл с форматом VOC и покинуть программу. Окно выбора файлов этой программы представлено на рис. 6.

Фактически, данная программа является оболочкой, использующей несколько различных программ и драйверов для работы с файлами различных форматов. К сожалению, возможны случаи конфликта драйверов, так что при работе с файлами разных форматов поведение данной программы оказывается не вполне устойчивым.

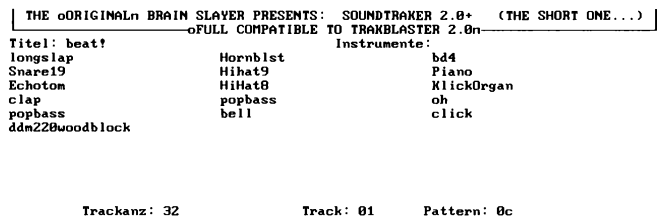


Рис. 3. Переднее панно простого проигрывателя MOD-файлов Sound Traker2.0

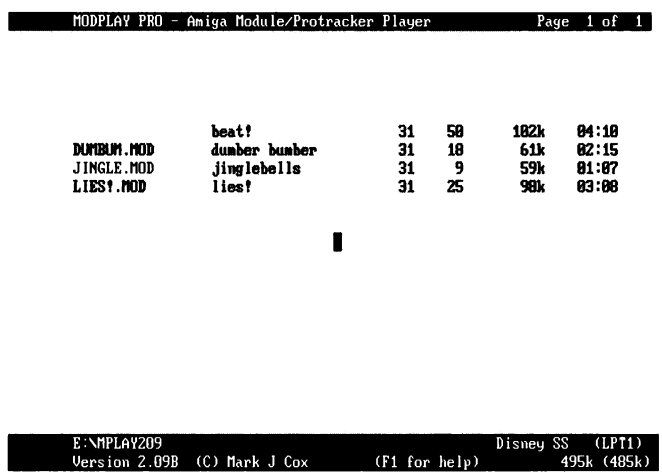


Рис. 4. Диалоговое окно загрузки файлов программы ModPlay Pro

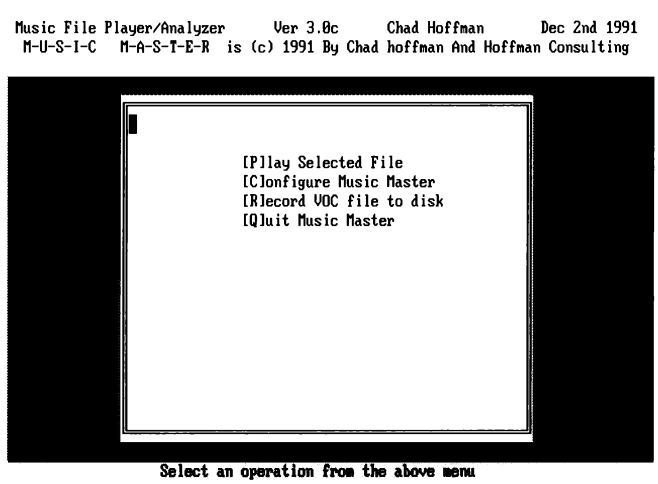


Рис. 5. Окно программы Music с главным меню

Достаточно современным интерфейсом обладает программа Voice Blaster 1.1 (рис. 7). Эта программа служит для проигрывания файлов с расширениями VOC и CMF. Удобное диалоговое окно позволяет легко находить и запускать на проигрывание нужные файлы.

Помимо программ воспроизведения аудиофайлов существует множество программ для их записи. Записываемый сигнал формируется обычно резидентным или временно включаемым микшером и может состоять из сигналов микрофона, линии, синтезируемых звуковой платой музыкальных звуков и голосов.

Простейшие программы записи звука записывают звуки (не синтезированные) на диск и создают файл с указанным именем. Запись идет до тех пор, пока пользователь не прервет ее — обычно нажатием клавиши Esc. В состав программного обеспечения звуковой платы Audio Excel Pro 16 входят следующие программы для записи звуковых файлов:

MCPRECW.EXE — запись звуковых файлов с расширением WAV,

MCPREGV.EXE — запись звуковых файлов с расширением VOC.

Помимо указания имени файла, в который идет запись, в эти программы можно ввести ряд ключей, задающих различные параметры — например, стерео или монозвучание, включение или отмена вывода сообщений об ошибках и прочее. Мы не приводим форматы записей ключей, поскольку заполучивший такие программы пользователь имеет информацию о них «из первых рук».

Примером программы с изысканным пользовательским интерфейсом является программа Blaster Master Pro 4.8, позволяющая воспроизводить и записывать файлы с расширениями VOC и CMF. Пользовательский интерфейс программы имитирует работу с проигрывателем, имеющим кнопочное управление (рис. 8). Работа с такой программой понятна любому пользователю.

Мы рассмотрели лишь небольшое число программ для воспроизведения и записи аудиофайлов. Лишь на одном лазерном диске «14000 Programs» можно найти десятки программ такого рода.

Преобразование форматов звуковых файлов

Процесс записи звуков довольно трудоемкий, и часто его приходится многократно повторять. Может оказаться, к примеру, что фоновая музыка забивает ваш голос, или записался рев акустического самовозбуждения, или из-за высокого уровня звука недопустимо велики нелинейные искажения. Если у вас неплохо получилась запись в одном формате, а вам понадобился другой, не обязательно повторять весь мучительный процесс записи для другого формата. Для преобразования из одного формата в другой служат специальные программы, например:

VOC2WAV.EXE — преобразует файл с расширением VOC в файл с расширением WAV,

WAV2VOC.EXE — преобразует файл с расширением WAV в файл с расширением VOC.

Оркестровка музыкальных произведений

У всех пользователей ПК неизгладимое впечатление оставляют программы, ориентированные на оркестровку музыкальных произведений. Даже простейшие из таких программ представляют профессиональный интерес для молодых композиторов и просто любителей музыки. Они могут сочинять произведе-

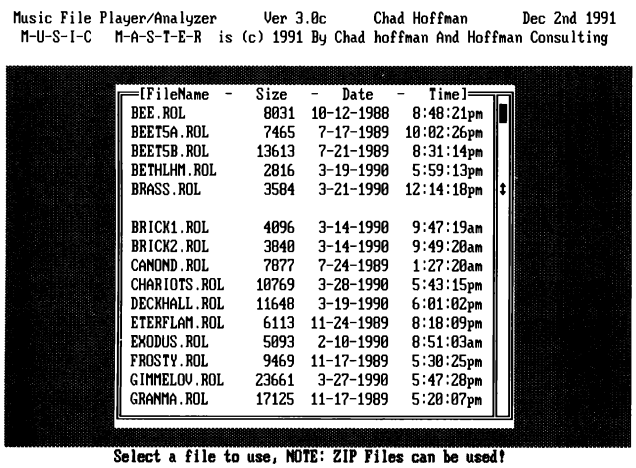


Рис. 6. Окно выбора файлов программы Music

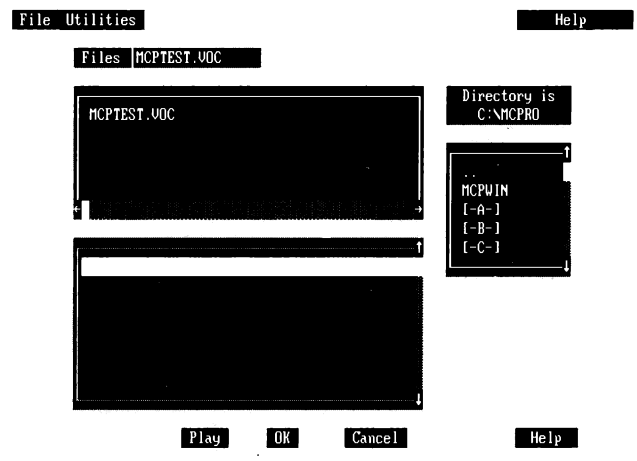


Рис. 7. Диалоговое окно программы Voice Blaster 1.1

ния и затем выполнять их оркестровку, записав нотами звучание каждого инструмента. Программа воспроизводит в стереорежиме звучание по заданным нотам. Такие программы восхищают даже тех, кто не имеет музыкального образования — для них в программы включается множество заранее подготовленных примеров. Их звучание наглядно свидетельствует о высоком музыкальном мастерстве компьютерного исполнителя музыки. Он не в состоянии соврать ни на какой ноте!

Одна из таких программ «Music» позволяет пользователю нанести на нотные линейки ноты для примерно двух десятков инструментов и тут же исполнить получившееся произведение. Программа имеет прекрасный пользовательский интерфейс (рис. 9). Большую часть переднего панно программы составляет окно редактирования нот. В верхней части окна расположено главное меню с выпадающим подменю. Ноты устанавливаются цифровыми клавишами, и их можно двигать вверх и вниз клавишами перемещения курсора. В целом, программа является прекрасным редактором для нот и позволяет распечатать ноты принтером (как матричным, так и лазерным).

Те, кто не разбирается в нотной азбуке, все равно восхищаются этой программой — к их услугам свыше десятка готовых музыкальных произведений, демонстрирующих профессионально выполненную оркестровку ряда популярных сочинений. Можно проигрывать как все выбранное произведение, так и его отмеченные фрагменты. Программа использует специальный формат музыкальных файлов с расширением NOT.

При подготовке музыкальных произведений, например клипов, часто бывает необходимо создавать различные звуковые эффекты — например, звуки ударных инструментов, возгласы, звуки выстрелов и так далее. Для этого тоже существует великое множество программ. Одной из таких программ является программа Drum Blaster, синтезирующая как одиночные, так и ритмично повторяющиеся звуки ударных инструментов. Панель управления этой программы показана на рис. 10 и не нуждается в особом комментарии.

Эта программа позволяет создавать более 100 ударных звуков. Вы можете легко стать ударником своего оркестра — домашнего или на дискотеке. Программу можно использовать и для задания ритма своих произведений. Приятно отметить естественность звучания ударных инструментов.

Управление средствами мультимедиа из MS-DOS программ

Мы не будем описывать здесь программирование плат звука на нижнем уровне. Оно необходимо для создания программ, подобных описанным выше. Естественно, что разработка таких программ требует высочайшей квалификации программистов и не под силу рядовым пользователям ПК. Но и целесообразность создания таких программ сейчас весьма сомнительна, поскольку их уже создано великое множество. Для большинства пользователей важнее научиться обращаться к таким программам прямо из своих программ, написанных на популярных языках программирования — от QBasic до C++.

Почти все языки программирования имеют средства для общения с внешними программами. К примеру, на QBasic таким средством является оператор SHELL Имя_файла. Он просто дает выход в

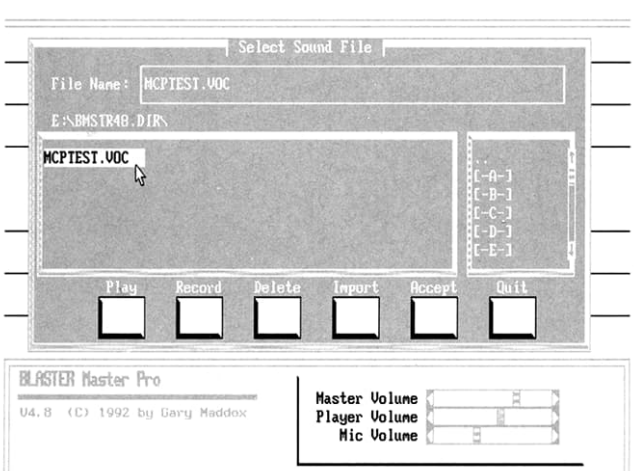


Рис. 8. Панель управления программой Blaster Master Pro 4.8

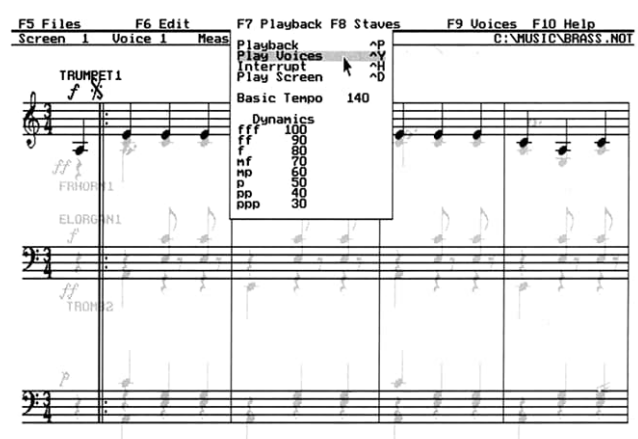


Рис. 9. Переднее панно синтезатора музыки по нотам

MS-DOS, исполняет файл с указанным именем и возвращается в головную QBasic-программу. Имя надо задавать по правилам MS-DOS, то есть в общем случае с указанием пути доступа к файлу. Это не нужно, если все файлы находятся в одном каталоге с программой. В состав имени могут входить параметры и ключи, определяющие характер работы исполняемого файла.

Зададимся целью создать программу, которая: просит пользователя загрузить лазерный аудиодиск в CD-ROM, выводит изображение проигрывателя таких дисков, позволяет пустить его в фоновом режиме, исполняет 10 секунд музыку с малой громкостью, затем 10 секунд исполняет музыку с большой громкостью, затем позволяет (с диалогом) записать ваш голос на фоне музыки и, наконец, прекращает работу, подав сигнал на выброс (Eject) кассеты с диском из проигрывателя.

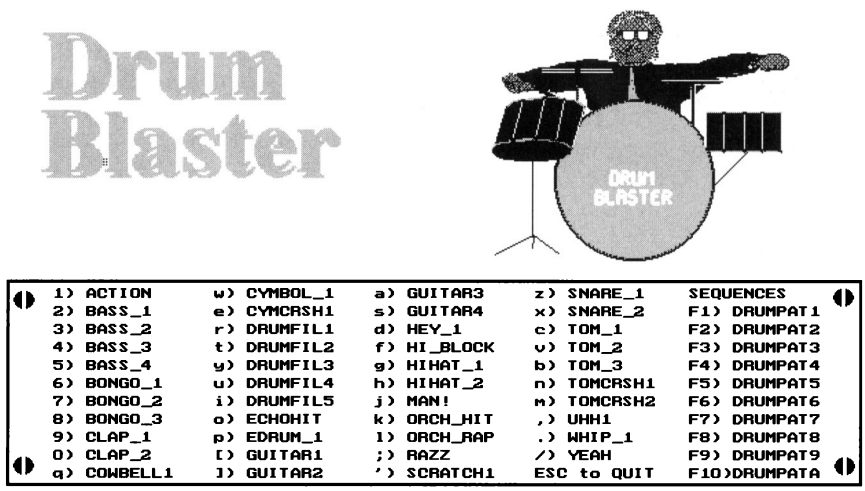


Рис. 10. Переднее панно синтезатора звуков ударных инструментов

Если вы внимательно прочитали о назначении программ управления CD-ROM и звуковой платой, описанных в этой статье, то приведенная в листинге 3 программа на QBasic не потребует других комментариев, кроме тех, что в ней есть:

Листинг 3

```
CLS
PRINT "Insert compact disk, press any key,"
PRINT "run CD-Player and set Power (off)"
F$ = INPUT$(1)
SHELL "C:\MCPRO\mcpsvol /C:5"  Set CD-volume (level 5)
SHELL "C:\CDROM\gsaudio.exe" 'Run CD-player
CLS   PRINT "10 Seconds Play Small Sound!"
SLEEP 10
SHELL "C:\MCPRO\mcpsvol /C:12" 'Set CD-volume (level 12)
CLS   PRINT "10 Second Play Big Sound!"
SLEEP 10
CLS   PRINT "For setting Mixer"
PRINT "press any key"
F$ = INPUT$(1)
SHELL "C:\MCPRO\mcp-mix"        'Set Mixer Program
SHELL                           'Exit in MS-DOS
SHELL "C:\MCPRO\mcp-mix -f"     'To remote from memory
CLS : PRINT "For save sound from microphon press any key!"
PRINT "After saving (no more 25-30 seconds) press key Esc"
KILL "demo.wav"                  'Kill file demo.wav
F$ = INPUT$(1)
SHELL "C:\MCPRO\mcprecw demo.wav"  'Record sound
SHELL "C:\CDROM\eject.exe"       'Eject compact disk
PRINT "Press any key for playing record"
F$ = INPUT$(1)
SHELL "C:\MCPRO\mcpplayw demo.wav" 'Play sound
CLS   PRINT "Press any key for end session!"
```

При работе с этой программой надо следовать не только комментариям, заданным в ней, но и тем, которые использованы в других программных средствах. К примеру, после выхода в MS-DOS для работы с микшером, после установки уровней звучания речи и фоновой музыки, нужно дать команду EXIT для возврата в программу. Это и есть зачатки интерактивного управления.

Итак, мы рассмотрели работу с наиболее распространенными прикладными программами для звуковых приложений мультимедиа, ориентированными на MS-DOS. Еще большие возможности, в том числе связанные с видео приложениями, открываются при использовании средств мультимедиа в графической оболочке Microsoft Windows. Однако это тема уже для других статей. ■

Литература

1. **В. Дьяконов. Я и мультимедиа!// Монитор-Аспект.– 1994.– № 2.**

Живое видео на экране компьютера

Д. Мокин

Мультимедиа возможности современных персональных компьютеров расширяются с каждым месяцем, мощные процессоры и видеосистемы делают доступным все больший спектр изобразительных средств. Но одна неотъемлемая часть мультимедиа до сих пор оставалась малодоступной для рядового потребителя. Речь идет о реалистичном видеоизображении. Искушенного читателя удивит это заявление: а как же системы нелинейного видеомонтажа, рекламируемые во всех изданиях? А как же серия Macintosh AV со встроенными средствами обработки видео? Но дело в том, что профессиональные системы нелинейного монтажа и создания компьютерных эффектов предназначены для работы с материалами длиной в несколько минут, да и базируются в основном на мощных рабочих станциях типа Silicon Graphics. А в Macintosh AV видеоролики QuickTime либо демонстрируются в окошке размером с почтовую марку, либо кадры меняются с частотой, весьма далекой от реалистичного восприятия.

Стандарт MPEG

Считается, что при представлении в компьютерной форме видеосигнала качества VHS достаточно разрешение 352 × 240 точек из 32 тысяч цветов. При этом одна секунда видеоинформации займет на диске около 4,5 Мб, 1 минута — около 290 Мб. К тому же, для работы с такими материалами нужны колоссального объема диски, имеющие вдобавок (как и компьютеры, в которых они установлены) огромную скорость передачи данных (если, конечно, вы хотите записывать и просматривать непрерывный видеофрагмент, а не отдельные кадры, иными словами, работать с видеоинформацией в реальном времени — Real Time). Единственный способ обрабатывать видеоинформацию в реальном времени — прибегать к процедурам сжатия. Причем сжатие должно производиться «на лету». Это значит, что видеосигнал при входе в компьютер должен упаковываться до попадания на жесткий диск, а еще лучше — до попадания на шину. И наоборот, при выводе «внутри компьютера» должен передаваться только сжатый материал, распаковка должна происходить как можно позже. Если алгоритм упаковки будет и эффективным, и быстрым, мы получим возможность хранить видео в компьютере и воспроизводить его в реальном времени. Такой способ упаковки видеоинформации «на лету» без потери качества известен как индустриальный стандарт MPEG-1. Он использует очень эффективный алгоритм упаковки, который позволяет сжать видеоданные качества VHS или Video-8 (вместе с сопровождающим стереозвуком CD качества) примерно в 30 раз. Это значит, что на обычном жестком диске емкостью 420 Мб можно хранить примерно 45 минут видеоматериалов, а на компакт-диске — фильм в 75 минут. Причем работа с источниками видеосигнала более высокого качества, S-VHS или Hi-8, показывает, что потеря качества и в этом случае практически незаметна. Алгоритм MPEG-сжатия существенно использует структуру видеоизображения, в котором последовательные кадры, как правило, имеют не очень сильные отличия. В MPEG-видео выделяются кадры трех типов: опорные кадры, содержащие полную картинку и используемые для получения произвольного доступа сразу к середине ролика, межопорные, которые могут быть построены на основе ближайшего опорного, и обычные, для восстановления картинки в которых необходима информация обо всех соседних кадрах. Благодаря такой сложной системе и достигается высокая эффективность упаковки и большая скорость воспроизведения. Правда, платой за это оказывается очень высокая вычислительная мощность, требуемая от системы, выполняющей сжатие. На сегодняшний день MPEG-стандарт уже активно используется производителями мультимедиа программ и видеопродукции. Ведущие фирмы уже выпускают фильмы на компакт-дисках в стандарте VideoCD, а в игры на компакт-дисках включают MPEG-видеофрагменты. Разработанный фирмой «Philips» стандарт CD-I (интерактивный компакт-диск) также опирается на использование MPEG-компрессии.

Техника

Еще недавно работа с MPEG была прерогативой мощных графических станций ценой в несколько сот тысяч

Статья публикуется на коммерческой основе.

долларов. Однако два современных продукта — **MPEGLab PRO** фирмы «Optibase» и **RealMagic** фирмы «Sigma Designs» открыли путь живому видео на PC. MPEGLab PRO — это набор из двух плат, вставляемых в мощный (не менее 486/33 МГц CPU, 8 Мб RAM, 600 Мб HDD) персональный компьютер. На платах установлены две микросхемы CL-4000 фирмы «C-cube». Эта микросхема — самостоятельный RISC-процессор, специально оптимизированный для выполнения MPEG-компрессии и приспособленный для параллельной обработки. Базовый PC играет здесь лишь вспомогательную роль, все равно даже Pentium или PowerPC не обеспечат достаточной скорости сжатия. Оснащенный системой MPEGLab PRO компьютер позволяет в реальном времени записывать видео и звуковое стереосопровождение на жесткий диск и впоследствии воспроизводить его. Стоимость компьютера, оснащенного системой MPEGLab PRO, — около 30 тысяч долларов. Это по крайней мере в 5 раз дешевле, чем графическая рабочая станция на MIPS-процессоре для работы с MPEG-видео. RealMagic (старое название ReelMagic) — это плата, которая выполняет только декомпрессию MPEG-видео и воспроизводит его на дисплее персонального компьютера. Для установки платы RealMagic требуется довольно скромный компьютер (386/25 МГц CPU, 2 Мб RAM, SVGA дисплей и привод CD-ROM). При этом она обеспечивает воспроизведение видео с частотой от 24 до 30 кадров в секунду (в зависимости от стандарта сигнала — PAL или NTSC) в окне на экране размером от 16 × 16 до 1024 × 768 точек и воспроизведение стереозвука с качеством CD. Специалисты знают: ни для PC, ни для Macintosh нет другого продукта с подобными свойствами. Стоит плата RealMagic довольно дешево — менее 500 долларов. Стоит упомянуть еще два типа технических устройств, связанных с цифровым видео. Во-первых, это накопители на магнитооптических (MO) дисках, уже знакомые многим пользователям. MO накопитель позволяет записывать на один сменный диск до 1300 Мб информации, а таких дисков вы можете иметь сколько угодно. Существуют так же системы с автоматической сменой и одновременной обработкой нескольких MO дисков. Такие системы, называемые JukeBox, могут поддерживать массивы информации до 180 гигабайтов! Во-вторых, это аппараты для записи компакт-дисков. Имея такой аппарат, вы сможете сами изготавливать малыми тиражами собственные компакт-диски, не прибегая к услугам заводов. Это может быть компакт-диск любого типа, с аудио, программной или MPEG-видео информацией.

Применение

Возможность хранить и обрабатывать в компьютере видеоинформацию, удовлетворяющую промышленным стандартам, настолько нова и непривычна, что все области применения трудно себе представить. Я постараюсь обрисовать некоторые из них, с тем, чтобы помочь читателям лучше представить себе открывающиеся возможности. Представим себе работу организации, в которой многие люди должны иметь свободный доступ к множеству видеосюжетов. Это может быть, например, агентство новостей, правительственная администрация или дорогой отель с кабельной видеосетью в номерах. В этом случае может быть построена компьютерная сеть, в центре которой будет находиться мощный видеосервер с системой JukeBox или большим массивом жестких дисков и станция ввода видеоизображений, оснащенная MPEGLab PRO. А каждый клиент сети с компьютером, оснащенным платой RealMagic, сможет в любой момент увидеть на своем дисплее любой сюжет из банка видеоданных на сервере. Более того, современные программные средства позволяют легко интегрировать видеоизображения в общую базу данных, связать их с сопутствующей информацией. Другое применение MPEG-видео связано с телекоммуникациями. На сегодняшний день существует только один способа передать видеоинформацию на большое расстояние: направить курьера с видеокассетой. Однако при использовании современных высокоскоростных модемов становится вполне реальной передача видеоматериалов в виде MPEG-файлов по обыкновенным телефонным сетям. Для кабельных студий и студий местного телевидения весьма удобно использовать в виде MPEG-файлов заставки, рекламные ролики и другие повторяющиеся сюжеты. Возможность мгновенно выбрать файл с требуемым материалом позволит не занимать несколько студийных магнитофонов и обеспечит выпускающему редактору намного большую свободу действий. Для тех, кто связан с мультимедиа и с интерактивным видео, преимущества использования MPEG в игровых, обучающих, справочных программах очевидны и не требуют объяснения. Повторим еще раз: перечислить и даже представить все области применения цифрового видео сегодня очень сложно, фантазия каждого программиста и пользователя может открыть новую область.

Практика

Оборудование для работы с MPEG-видео на российский рынок поставляет фирма **YAM International**, являющаяся официальным представителем фирм «Optibase» и «SigmaDesign». В этой фирме вы сможете получить подробные консультации по вопросам, затронутым в этой статье, а также увидеть в действии соответствующее оборудование. Мы понимаем, что если приобретение платы RealMagic или привода MO дисков может себе позволить практически любая организация, связанная с мультимедиа или с видео, то покупка системы MPEGLab PRO — серьезный шаг, на который не всякая фирма может решиться сразу. Однако та же фирма YAM International предоставляет своим клиентам сервис по переводу видеоматериалов с кассет в MPEG-формат. Надеемся, что наша статья пробудит ваш интерес к современным технологиям цифрового видео и поможет представить, как их можно использовать в вашей деятельности. ■

КОНТАКТЫ

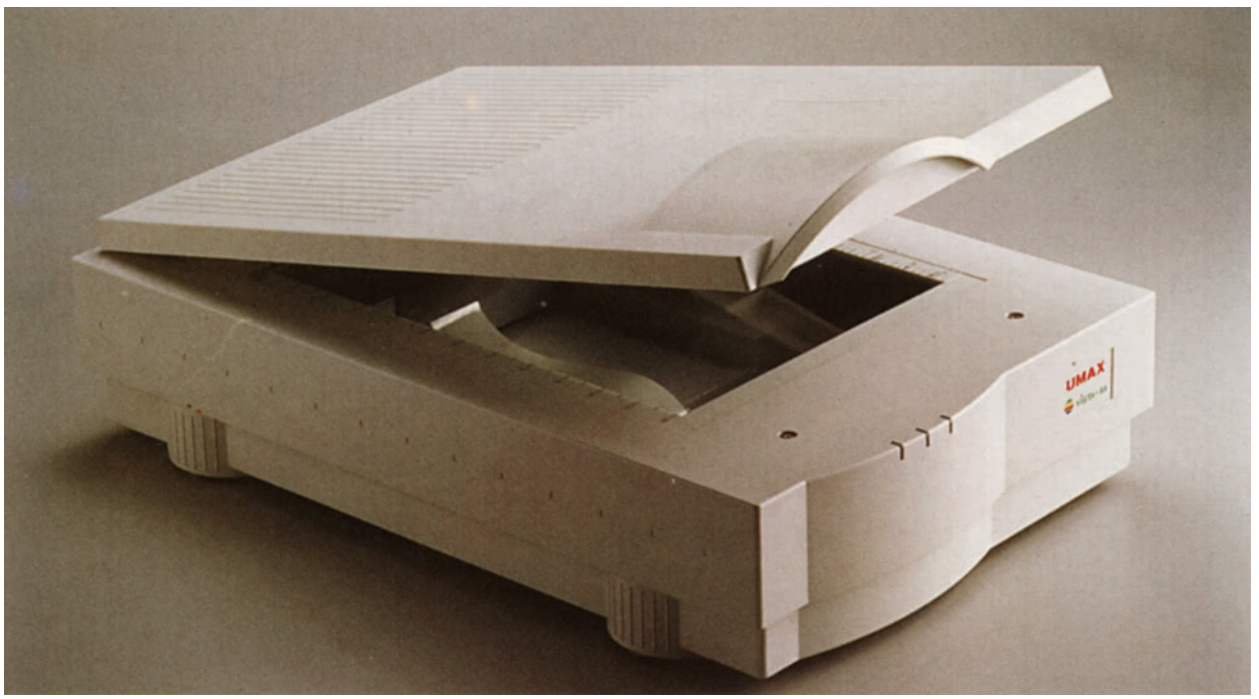
YAM International

Москва,
Профсоюзная ул.,57,к. 601.
Телефон: 232-64-20.

UMAX

ЛУЧШИЕ В МИРЕ ЦВЕТНЫЕ ПЛАНШЕТНЫЕ СКАНЕРЫ

ИХ ОФИЦИАЛЬНЫЙ ДИСТРИБУТОР В СНГ



Коллекция новейших моделей:

- Vista-T630
  Экономичный цветной сканер для офисов. Скорость - до 13 страниц А4 в минуту.
- Vista-S6 и S8
  Идеальные сканеры для настольных издательских систем. Считывание в один проход.
- PowerLook
  Профессиональный сканер для издателей. 30 бит. Диапазон оптических плотностей - 3.0. Включает приставку для сканирования слайдов.

Признанные наиболее удобными для пользователей пакеты сканирования,калибровки цвета и обработки изображений:

- MagicScan
- MagicMatch
- PhotoShop LE/FE

По желанию: устройство автоматической подачи оригиналов,приставка для слайдов.

Цены - начиная с $895. Поставка со склада в Москве.

YAM International:
Ул. Профсоюзная,57,Москва.
Тел.: 334-23-44. Факс: 334-28-01.

Приглашаем дилеров!

«Hero Quest III» — путь к славе

Избавьте страну ваших друзей от демонов и восстановите мир!

Д. Стариков

PC

Ad hominem

Название игры:	*«Hero Quest III»*
Фирма-разработчик:	*«Sierra On-Line»*
Звук:	*PC Speaker/Tandy/Sound Blaster/Adlib*
Графика:	*EGA/VGA*
Управление:	*клавиатура/джойстик/«мышь»*

«Hero Quest III» — продолжение серии игр с одноименным названием. Она имеет прекрасный колоритный сюжет, проще в управлении, ярче и красочнее, нежели предшествующие игры. Здесь вас тоже поджидают опасности, коварные враги, монстры-обитатели саванны и джунглей, злая магия демонов. Поэтому во время игры вам скучать не придется. Смекалка и хитрость во многом помогут вам на протяжении всей игры.

Сюжет игры Mar (Wizard)

Восстановив справедливость и избавив два города от сил зла, а также покончив со злым магом Адом Ависом (Ad Avis), вы попадаете на прием к султану, принимаете похвалы и поздравления. Вашим друзьям Ухаре (Uhura) и Ракишу (Rakeesh) надо возвратиться домой, так как там появились злые демоны, да и распри между племенами страны надо остановить. И вы как истинный герой отправляетесь вместе с ними на поиски новых приключений и побед.

Жена Ракиша — Криша(Kreesha), обладая большими познаниями в области магии, переносит вас с помощью магического портала (portal) в столицу страны человекольвов (Liontaur) — город Тарну (Tarna). Ухара с ребенком, отправляется в деревню своего племени Симбани (Simbani), а вы, поговорив со счастливыми от встречи Ракишем и Кришей, идете на обследование города. Здесь есть: аптека (apothecary), гостиница (inn), дом Ракиша и Криши, суд (Hall of Judgement), тронный зал (King's Chambers), храм богини человекольвов (Temple of Sekhmet), но главной достопримечательностью города является базар (bazar). Первым делом вам надо обменять свои деньги на местную валюту (royals, commons), чтобы приобретать нужные вещи или лекарства. Благодаря случайной встрече с вором Харами (Harami) вам выпадет случай познакомиться с работой Судебного совета (Council of Judgement) и поговорить с самим раджей (Rajah Sah Tarna), братом Ракиша, от него вы узнаете о неудачной попытке уничтожения демонов, предпринятой дочерью Ракиша и Криши Ришакой (Reeshaka) и смельчаком по имени Хатиб Мукаррам (Khatib Mukar'ram), а также о возложенной на Ракиша миссии мира и поиска дочери. От аптекаря Салима (Salim Nafs) вы узнаете, какие ингредиенты нужны для магического снадобья (dispel potion) и исцеляющих пилюль (healing pills) и где их отыскать. В гостинице на доске объявлений вы увидите законы (laws) и объявления. У девушки-официантки Жанны (Janna Jamil) узнаете множество интересного о городе и его жителях. Кроме того,

вам предоставят комнату для отдыха.

На базаре вы встретите торговца сувенирами Шаллаха (Shallah), принадлежащего к народности Катта (Katta). Зайдите в храм, и сама богиня Сехмет (Sekhmet) поговорит с вами и даст поручение. Теперь отправляйтесь прогуляться по саванне. Отыскав место, где на камнях живут летучие мыши (meerbat), и оказав им помощь, вы получите награду. Вскоре вы вместе с Ракишем отправитесь в деревню скотоводов-кочевников Симбани для того, чтобы предотвратить войну между Симбани и племенем людей-леопардов (Leopardmen). Разобравшись в сути конфликта, вы пообещаете Ракишу все уладить, и ваш друг возвратится в Тарну, искренне надеясь на вашу помощь. Проснувшись утром следующего дня, вы начинаете знакомиться с деревней. Вождь племени — Лайбион (Laibion) расскажет о реальных причинах конфликта. Его сын Есуфу(Yesufu) тоже кое-что расскажет и научит играть в увлекательную игру Авари (Awari). Можете попробовать потренироваться на специфических тренажерах Симбани.

Иногда в деревне странствующий сказитель декламирует увлекательные баллады. Наедине вы сможете расспросить его о подробностях. Выйдя из деревни, отправляйтесь на исследование окрестностей. Наберите в озере воды в бурдюк (waterskin) и отправляйтесь к большому дереву под названием — «Матерь Мира» (Mother of World), о котором вы узнаете от Салима. Здесь вы знакомитесь с невидимыми Стражами (guardian) этого волшебного дерева (Magic Tree), рассказываете о своих победах и получаете некоторые необходимые вещи. На границе саванны с джунглями вы отыскиваете веточку голубой орхидеи (blue orchid) и с помощью магии удивительного дерева делаете ее волшебной. Далее на очереди исследование джунглей. Побродив по берегу водопада (waterfall), вы обнаружите пойманную в ловушку обезьянку Ману (Manu). Освободив ее и поговорив с ней, приобретете нового друга.

Возвратившись в деревню, вы увидите, что в специальной клетке находится человек-леопард. Ухар расскажет вам о том, как пленник был захвачен и как заставить его говорить. Вы возвращаетесь в Тарну, отдаете аптекарю собранные во время путешествия компоненты и на следующий день получаете магический эликсир. Пока Салим готовит его, вы должны научиться пользоваться волшебной веткой, Криша с блеском обучит вас этому. Полученное от аптекаря снадобье поможет вам расколдовать пленника, и тот превратится в очаровательную девушку по имени Джохар (Johar), дочь вождя людей-леопардов. Есуфу, сын вождя, полюбит ее и захочет, предоставив выкуп, жениться на ней, но отец не позволит ему. Вы как истинный джентльмен внесете выкуп и освободите пленницу, которая сразу же убежит прочь из деревни. Отправляйтесь на поиски и обязательно встретите ее, но, поговорив с вами, она опять скроется. Так будет несколько раз, пока вы не убедите ее в необходимости предотвратить кровавый конфликт между двумя племенами. Поверив вам, девушка откровенно ответит на все вопросы, научит магии и отведет в свою заколдованную деревню людей-леопардов. Там вы увидите обряд превращения людей в леопардов. Вождь племени вашей спутницы примет вас с радостью, потому что обязан вам свободой дочери. Но шаман (Shaman), оказавшийся приспешником демонов, вынудит вас к магической дуэли (magic duel). Одержав победу в трудном поединке, вы завоюете дружбу людей-леопардов и, возвратив враждующим племенам украденные реликвии — их священные символы, восстановите мир и согласие. Но во время проходящей в судебном совете Тарны конференции мира вождь людей-леопардов, одержимый властью демонов, убивает вождя Симбани, а Есуфу, присутствующий там, умерщвляет вождя людей-леопардов. Раджа приходит в бешенство от разыгравшейся на его глазах трагедии, а вы по совету мудрого Ракиша спешно покидаете столицу, чтобы проникнуть в погибший город (lost city) и уничтожить ненавистных демонов прямо в их логове.

Теперь перед вами стоит последняя, самая трудная и опасная задача — избавить народы страны от коварных демонов. Ракиш объяснит, что благодаря своей магической силе Криша узнала, что демоны скрываются в погибшем городе, расположенном на другой стороне водопада. Выслушав наставления, вы немедленно отправляетесь в путь. По дороге, на привале, вы внезапно встречаетесь с представителем маленького народца земных свинок (earth pigs) по имени Арне Сакнусен (Arne Saknoosen). Этот обаятельный собеседник рассказывает вам много интересного о стране человеко-львов и о населяющих ее народах, о своем племени и о погибшем городе, куда вы держите путь.

После этого вас ждет еще одна приятная встреча с сердечно признательной вам обезьянкой Ману, и вы отправляетесь в гости к племени обезьян. Благодаря этой неожиданной встрече вы многое узнаете о погибшем городе. Обезьяны отговаривают вас от похода туда, но вы настаиваете на своем, и они отступают и соглашаются провести по единственной дороге к проклятому, по их мнению, месту. Преодолев по пути некоторые препятствия, вы добираетесь до места, где обезьяны покидают вас.

Пройдя незаметно через потайную дверь, вы попадаете в боль-

шой зал и становитесь свидетелем разговора двух огромных страшных демонов. В конце концов они обнаруживают постороннего, и вы вступаете в схватку с одним из чудовищ. Выйдя победителем, вы отправляетесь на поиски второго демона, а сталкиваетесь лицом к лицу с пропавшей Ришакой. Но это хитрая ловушка, и вам опять приходится драться с демоном. Победив и на этот раз, вы встречаете нескольких друзей и знакомых, которые помогут в поединке с оборотнями. Вор Харами заменит вас в бою и даст возможность остановить могущественного демона-мага (demon wizard), пытающегося вызвать волшебную колесницу, и если вам не удастся помешать этому, то прекрасная страна друзей навсегда попадет под власть кровожадных чудовищ. Решающая схватка происходит на вершине старой башни. Проявив истинное мастерство мага, вы одерживаете блестящую победу и избавляете наконец эту чудесную страну от власти злых демонов. Благодарные друзья торжественно поздравляют вас, от души радуясь победе.

Но темные силы не дремлют, и зло, бесчинствующее в глуши лесных дебрей, уже пробует сломить вас. Теперь вы знаете, что заканчивать свое победоносное шествие в битве с врагами рано и что где-то ждут вашей помощи и защиты.

Некоторые советы

Откройте подарок от продавца магазина магии Кипона Лаффина (Keapon Laffin), известного вам по второй игре, и, прочитав текст на оберточной бумаге, вы овладеете тайной фокуса с освещением (juggling lights spell). На базаре после обмена валюты вам надо купить трутницу (tinder box), тонкое копье (fine spear), баночку меда (honey), пять шкур зебры (zebra skin), бурдюк (waterskin), веревку (rope), бусы (string of beads) и красивый восточный халат (fine robe). Отдайте Катте письмо от Шимы (Shema), знакомой по «HQ II», и в знак дружбы получите в подарок фигурку леопарда (wooden leopard). На оставшиеся деньги можете приобрести что угодно. Для изготовления магического снадобья (dispel potion) необходимы следующие компоненты: вода (water), ядовитый виноград (venomous vine fruit) и подарок, сделанный от всего сердца (Gift from the Heart).

Воду вы сможете набрать в пруду (Pool of Peace), расположенном недалеко от деревни Симбани. Для того чтобы получить ядовитый виноград, надо отыскать в саванне место, где обитают летучие мыши (meerbat). Вы уйдете оттуда, но потом вернетесь и увидите, что зверьки учат летать малыша, который попадает в плен к растущим здесь кустарникам. Бросайтесь на выручку, а запутавшись в ветвях, вынимайте кинжал и рубите. Неожиданно растения, отпустив свою добычу, исчезают в скалистом грунте. Снова уходите, а вернувшись, обнаружите на скале возле нор зверьков подарки — ядовитый виноград и драгоценный камень (fire opal). Последний компонент для изготовления снадобья вы получите на волшебном дереве «Матерь Мира». Попав на площадку, где в волшебном кругу переливаются фиолетовым и желтым цветом звездочки, дайте им завладеть собой. Это стражники дерева, которые дадут вам любую нужную информацию (о подарке сердца, о гемме и о волшебной ветке). Следуя их совету, поднимитесь на верхнюю площадку, в самое сердце дерева, и, очутившись перед винтовым возвышением, войдите и полейте растение водой из пруда. На ветке сразу же появится нужный вам подарок. Поручение богини Сехмет вы исполните, попросив нужную вам гемму Стражей (Gem of Guardian) у самих стражников волшебного дерева «Матерь Мира». Вы отыщете ветку голубой орхидеи (blue orchid), побродив по границе саванны и джунглей (две темные полосы), произнесите заклинание (fetch spell) и сорвите ее с дерева. В полночь окуните ветвь в пруд, возвратитесь в сердце волшебного дерева и положите ветвь на винтовое возвышение. Теперь у вас есть магический посох (magic wood), но использовать его как одну из чар вас научит Криша.

Перо медовой птицы (feather of honey bird), которую вы встретите, можно добыть следующим образом: 1) дайте птице возможность довести вас до дерева с пчелами и выйдите; 2) налейте под дерево немного меда и снова уходите; 3) подойдите и, спугнув птицу, берите перо. В обмен на перо медовой птицы вы получите от аптекаря три исцеляющие пилюли (healing pills).

Лэйбиону вы заплатите выкуп: пять шкур зебры, тонкое копье и великолепный восточный халат — и можете брать пленницу в жены, а сделав ей по свадебному обычаю Симбани два подарка (фигурку леопарда и нитку бус), смягчите гордый характер пленной девушки. Едва вы, по праву мужа, откроете клетку, девушка, легко перемахнув через изгородь, убежит. Вы обязательно встретите ее на границе саванны и джунглей.

Согласившись на дуэль с шаманом, вы должны произнести заклинания (spell) в следующем порядке: 1) Magic Staff; 2) Reversal; 3) Calm (против огня); 4) Open (против клетки); 5) Jaggling lights (против темноты); 6) Dazzle (против иллюзии пояса); 7) Levitate (против пропасти). После этого вас ждет еще один, теперь уже смертельный поединок с разозлившимся шаманом, в котором и выступит наружу его истинная сущность приспешника демонов. Победив,

попросите у вождя людей-леопардов копье смерти (Speare of Death), являющееся святыней у народа Симбани. Обменяв копье смерти на магический барабан (Magic Drum), святыню людей-леопардов, вы восстановите мир между Симбани и людьми-леопардами. Чтобы забраться на деревья, где расположена деревня обезьян, произнесите заклинание levitate. Во время путешествия к развалинам старинного города, где обитают обезьяны, вам предстоит переправиться через глубокое ущелье. Для этого надо бросить один конец веревки на противоположную сторону обезьянке Ману и произнести заклинание levitate. Как только взлетите, Ману перетянет вас через пропасть.

Перед тем как войти в обитель демонов, следует хорошенько отдохнуть, чтобы полностью восстановить силы. Чтобы пройти через потайную дверь, надо вставить огненный опал (fire opal) в глазницу украшающей дверь скульптуры. После победы над первым демоном восстановите силу и магические способности с помощью пилюль и проходите в следующую дверь. Чтобы помешать демону вызвать магическую колесницу,а заодно и уничтожить его, надо использовать: Reversal, Calm (против огня), Trigger (против чудовища), Magic Staff, затем, как только ваш волшебный посох окажется в руках у демона, Trigger Spell и, наконец, Force Bolt spell, против расположенной с правой стороны небольшой белой державы (Orb), с помощью которой демон вызывал магическую колесницу.

Управление игрой

Управление осуществляется «мышью». Все необходимые команды располагаются в едином верхнем меню в квадратиках с изображением какого-либо действия или специального значка. Выбрав нужное действие, надо перенести его изображение в интересующее вас место или на персонаж и щелкнуть левой клавишей «мыши». Нажатием правой клавиши «мыши» вы можете выбрать без вызова верхнего меню только команды с изображением действия героя. Средняя клавиша «мыши» позволит перейти, минуя остальные команды, сразу к первой, и наоборот. Квадратик со знаком вопроса объяснит назначение любого другого. Выбрав разговор (изображение рта) и перенеся его на своего героя, вы можете отвечать на вопросы и рассказывать о чем-нибудь. Если вы бросаете предметы или используете магию, цель также наводится «мышью». Ну, вот и все. Отправляйтесь в удивительный мир игры, где добро всегда побеждает зло и ждут интересные приключения и занимательные встречи. Приятного вам времяпровождения. ■

EWCHI'95 ***THE 1995 EAST-WEST INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION***

«ВЗАИМОДЕЙСТВИЕ ЧЕЛОВЕКА С КОМПЬЮТЕРОМ»

МОСКВА, РОССИЯ, 4-8 июля 1995

Доклады и компьютерные демонстрации принимаются по самым разным проблемам человеко-компьютерного взаимодействия:

Компьютеры в образовании
Теория деятельности
Гипертекст и мультимедиа
Интерфейсы для инвалидов
Человеческие факторы и эргономика
Формальные методы и моделирование
Объектно-ориентированные языки, системы и среды
Базы знаний, экспертные системы и интерфейсы
Общение через компьютерные сети
Практика тестирования пользователей и эмпирические оценки
Стардартизация интерфейсов
Инструментальные системы
Психология программирования

Будут организованы также 4-часовые **учебные семинары (tutorials)** по наиболее актуальным HCI-проблемам.

Председатели конференции: **Bonnie Nardi** («Apple Computer, Inc.») **Юрий Горностаев** (МЦНТИ)

ДОКЛАДЫ И ЗАЯВКИ ОТ УЧАСТНИКОВ ИЗ РОССИИ, СТРАН СНГ И БАЛТИИ посылать по адресу:

Россия, Москва, 125252, ул. Куусинена, д. 21б, МЦНТИ.
Тел.: (095) 198-7691, 198-7041, 198-7350.
Факс: (095) 943-0089. E-mail: enir@ccic.icsti.msk.su

МИНИ-АТС 1x6

Мини-АТС GT-HS106
идеальна для дома
или небольшого офиса

цена - от $81 до $376 !!!

тел. (095) 325-0088, 322-0879;
факс (095) 325-2293
ЛАТЕКС

Отличное качество звучания
Великолепная эргономика
Оригинальный дизайн
Необычные функции

Изменение голоса, удаленное прослушивание

ТЕЛЕФОНЫ

Эталонная платформа для PowerPC®

7 ноября 1994 года «Apple Computer, Inc.», IBM Corporation и «Motorola» пришли к соглашению о создании новой компьютерной платформы для микропроцессоров PowerPC®. Новая платформа определяет архитектуру, которая может быть использована любым производителем аппаратных средств для создания компьютеров на базе PowerPC®, работающих с прикладными программами, написанными для различных операционных систем. Новая система будет поддерживать такие операционные системы, как MacOS™, OS/2™, AIX™ и Windows NT™. Более полная техническая информация по новой компьютерной платформе будет опубликована весной 1995 года. Появление новых компьютеров ожидается в 1996 году.

Слияние «U.S.Robotics» и «Megahertz»

19 октября 1994 года компания «U.S.Robotics», являющаяся одним из лидеров в области разработки, производства и маркетинга коммуникационных систем и продуктов, и «Megahertz» — поставщик мобильных коммуникационных продуктов и производитель средств связи на основе PCMCIA — подписали договор об окончательном слиянии.

«Intel» модернизирует семейство ProShare

В середине ноября 1994 года корпорация «Intel» объявила о расширении своего семейства средств персональной конференц-связи ProShare, включив в него средства видеоконференц-связи через локальные сети в дополнение к цифровым сетям ISDN. «Intel» также внедрила приставку к своему семейству систем управления LANDesk Manager, позволяющую администраторам сетей развертывать средства видеоконференц-связи через существующие корпоративные сети.

«Microsoft» объявляет о выходе Access Upsizing Tools

На состоявшейся в конце 1994 года конференции и выставке «Access Advisor» корпорация «Microsoft» объявила о выходе Microsoft® Access Upsizing Tools, дополнительного продукта для системы управления базами данных Microsoft® Access 2.0. Эти инструментальные средства позволят пользователям Microsoft® Access частично или полностью преобразовать или «нарастить» базу данных, основанную на Micosoft® Access в Microsoft® SQL Server™. Ключевыми механизмами продукта являются инструменты Upsizing Wizard, непосредственно позволяющие наращивать базы данных, и Browsing Wizard, обеспечивающие пользователям Access доступ к базе данных Microsoft® SQL Server™ из «клиент-сервер» контейнера, созданного в Microsoft® Access. Microsoft® Access Upsizing Tools можно приобрести примерно за 99 долларов США.

Комплексное решение для управления персональными компьютерами

В конце 1994 года корпорация «Microsoft» объявила о выходе Microsoft® Systems Management Server версии 1.0 (ранее известного под кодовым названием «Hermes») для операционной системы Windows NT™ версии 3.5. Эта система обеспечивает более простое и экономичное управление аппаратным и программным обеспечением на рабочих станциях через широкомасштабные компьютерные системы.

Microsoft® SNA Server 2.1 для Windows NT 3.5

Корпорация «Microsoft» представила в конце 1994 года Microsoft® SNA Server версии 2.1 для операционной системы Microsoft® Windows NT™ версии 3.5. Этот продукт значительно расширяет мощность, производительность и поддержку сетей, прикладных программ и аппаратных средств. Менеджеры информационных систем управления и конечные пользователи получают, наряду со многими другими преимуществами, еще и более простую и полную связь с IBM®.

Интегрированная информационная система BackOffice

Корпорация «Microsoft» представила новую интегрированную информационную систему BackOffice, облегчающую компаниям процесс принятия решений и стандартизацию деловых процессов. В систему входят операционная система Microsoft® Windows NT™ Server версии 3.5, система управления базой данных Microsoft® SQL Server™ версии 4.21a, сервер Microsoft® SNA Server 2.1, Microsoft® Systems Management Server 1.0 и Microsoft® Mail Server 3.2. Система объявлена как часть общей архитектуры «Microsoft» для распределенной обработки данных.

Компании объявляют о поддержке Microsoft® BackOffice

В связи с выходом нового семейства интегрированных серверных прикладных программ Microsoft® BackOffice корпорация «Microsoft» объявила о расширении отношений с тремя компаниями: «Compaq Computer Corporation», «Peoplesoft, Inc.» и SAP AG, которые будут обеспечивать расширенную поддержку системы управления базами данных Microsoft® SQL Server™ и платформы Microsoft® BackOffice. «Compaq» будет распространять системы во всем мире с каждым сервером COMPAQ® Proliant™, «Peoplesoft» будет поддерживать продукты в своих разработках PeopleTools® и Peoplesoft HRMS (система управления трудовыми ресурсами), а SAP планирует поддержку Microsoft® SQL Server в своем пакете R/3, объединяющем интегрированные промышленные прикладные программы для управления

финансами, трудовыми ресурсами и производством.

«Lotus» анонсирует Approach® версии 3.0

Корпорация «Lotus Development» объявила о выходе Lotus® Approach® 3.0 для Windows™, новой версии реляционной базы данных. В продукте воплощены более 200 новых возможностей, среди которых шаблоны баз данных, возможность изменения меню и задания макросов и многое другое. База данных может быть интегрирована в Lotus® 1-2-3, а также в Lotus Notes®. Для функционирования программы необходим 386-й IBM совместимый компьютер с 4 Мб памяти. В странах СНГ Approach® будет продаваться по 144 доллара (русская версия) и по 159 долларов (английская версия). Кроме этого, по заявлению компании, выход очередной версии совпал с распространением миллионной копии программы.

«Lotus» распространяет 1-2-3 версии 5 совместно с Approach® 3.0

«Lotus Development Corp.» представила специальную совместную версию Lotus® 1-2-3 для Windows™ и базы данных Approach® 3.0. Эти два новых приложения интегрированы друг в друга для обеспечения лучшей связи данных. Работая в Lotus® 1-2-3, пользователи нового пакета получат в меню Tool, Databases четыре новых возможности: Report, обеспечивающую доступ к Approach® Report Assistant, Form, позволяющую создавать формы посредством генератора форм Approach®, Dynamic Crosstab, анализирующую динамически изменяющиеся данные и Mailing labels, создающую почтовые этикетки по стандарту Avery. Для устойчивой работы пакета необходим 386 процессор и 6 Мб оперативной памяти.

«Borland» объявляет о новинках российского рынка

Компания «Borland» объявила в конце 1994 года о начале поставок на рынок стран СНГ и Прибалтики новых программных продуктов: Borland Database Engine для Windows™, dBASE 5.0 Distribution Pack для Windows™ и upgrade версии системы управления базами данных Paradox 5.0 для Windows™. С появлением Database Engine разработчики получили доступ к мощным возможностям интерфейса IDAPI. Ожидаемая розничная цена пакета 299 долларов. Новый продукт dBASE 5.0 Distribution Pack представляет собой runtime версию и лицензию для разработчика на бесплатное распространение runtime библиотек dBASE для Windows™ в составе коммерческого приложения. Ожидаемая цена — 340 долларов. Upgrade-версия системы Paradox представляет собой полнофункциональный пакет с документацией, продаваемый зарегистрированным пользователям по сниженной цене (139 долларов).

Новая среда разработки приложений «клиент-сервер» компании «Borland»

Компания «Borland» объявила о разработке новой высокопроизводительной среды для разработки приложений типа «клиент-сервер» с кодовым названием Delphi'95. Утверждается, что система станет уникальным дополнением к комплексу средств разработки таких приложений за счет множества новшеств, воплощенных в ней, о которых в связи с продолжающимся бета-тестированием не сообщается.

Новая система защиты Novex Disk

16 ноября 1994 года в рамках выставки «Банк и офис-94» состоялась презентация новой системы защиты информации для компьютеров notebook Novex Disk, представленной фирмой «Актив». Система состоит из уплотнителя данных на винчестере, работающего на основе оригинальной технологии компании «Roller, Ltd.» и сжимающего данные с коэффициентом 2:1, дополненного мощной системой парольной программной и аппаратной защиты. В качестве аппаратной защиты используются электронные таблетки типа touch-memory компании «DALLAS Semiconductor». На 286-х машинах программа занимает 51 кб обычной памяти при отсутствии EMS или XMS и 5 Кб при их наличии (остальная часть располагается в памяти указанного типа), а при работе на 386-х и более старших моделях компьютеров программа полностью размещается в верхней памяти. Ожидаемое время начала поставок по цене около 95 долларов — конец 1994 — начало 1995 года. В комплект будут входить 3 таблетки DALLAS: одна — с самым высоким уровнем доступа — для владельца компьютера и 2 для его знакомых или коллег, имеющих доступ к этому же компьютеру. В планах компании «Актив» создание специальной платы расширения для защиты компьютеров.

«Cabletron» представляет решение на основе Token Ring

В середине ноября 1994 года компания «Cabletron Systems» объявила о четырех новых продуктах Token Ring, разработанных для обеспечения полноуправляемых, легко расширяемых офисных решений. Новые продукты включают Cabletron MicroMMAC-T, первый разветвитель (hub) Token Ring для конечных филиалов, обеспечивающий SNA/LAN/WAN совместимость и расширенные возможности управления, поддерживающие все 10 уровней RMON-групп. Cabletron также представляет SNA/WAN интегрированный модуль для MicroMMAC вместе с двумя новыми стекируемыми хабами Token Ring из серии HubSTACK™ — STH и STHi.

Сотрудничество «Cabletron» и «AXON»

Компания «Cabletron Systems» в ноябре 1994 года объявила о присоединении «AXON Networks, Inc.» к программе партнеров-разработчи-

ков SPECTRUM. Компания «AXON» будет интегрировать свой набор LANServant с платформой управления сетями SPECTRUM для обеспечения менеджеров сетей возможностью анализировать загруженность удаленных сегментов сетей, в которых обычно нет ни управляемого программного обеспечения, ни квалифицированных инженеров.

«Novell» анонсирует новую программу марки UnixWare

В конце 1994 года «Novell» объявила о новой маркетинговой и сертификационной программе UnixWare Branding initiative. В качестве расширения высокоэффективной программы YES для NetWare, новая программа гарантирует покупателям расширяемость и совместимость с решениями UNIX enterprise. При поддержке более двадцати индустриальных партнеров программа марки UnixWare указывает на рост влияния системы как стандарта UNIX. Новая программа является открытым решением, так как облегчает пользователям возможность выбора из широкого набора обновленных приложений и платформ, поддерживаемых производителями оборудования и независимыми партнерами по программному обеспечению.

«Novell» анонсирует компилятор C++ для UnixWare

Разработчики приложений для UnixWare, давно ожидающие более производительное средство программирования, получили такую возможность с выпуском в конце 1994 года C++ Compilation System. Система позволяет разработчикам использовать новейшие технологии при разработке надежных приложений для платформы UnixWare 1.1. Система включает: высокопроизводительный компилятор C++, оптимизированный для процессора Pentium™; библиотеки стандартных компонентов C++; графический отладчик с настраиваемым интерфейсом и анимационными возможностями; системные и C++ заголовочные файлы; поддержку разделяемых библиотек. Система будет распространяться на CD-ROM по цене около 100 долларов.

«Novell» и «Sybase» представляют SybaseWare

Компании «Novell» и «Sybase» представили новое интегрированное решение «клиент-сервер», обеспечивающее лучшую в классе интеграцию баз данных и сетей. SybaseWare упрощает установку и управление, а также позволяет быстро и эффективно разрабатывать и внедрять системы «клиент-сервер». SybaseWare — полностью интегрированная сетевая платформа, состоящая из SYBASEWorkGroup SQL Server10 и операционной системы UnixWare. Также в состав продукта входят: SQL Server Monitor для Windows — графическое средство для оптимизации производительности SQL Сервера, SQL Server Manager для Windows — графический DBA-инструмент для SQL Сервера и BackUp Server — средство для параллельного выполнения архивирования (backup) и восстановления. Сегодня цена на систему колеблется от 995 долларов для одного пользователя до 19995 долларов за систему на 100 пользователей.

«Novell» анонсирует протокол NetWare Link Service

В ответ на повсеместно возросшие требования к коммуникациям локальных (LAN) и глобальных (WAN) сетей «Novell Inc.» в конце 1994 года представила протокол NetWare Link Services (NLSP) для пользователей серверов NetWare 3.x и 4.x. В сравнении с протоколами Routing Information и Service Advertising новый протокол уменьшает загруженность сетей более чем в 40 раз. Также NLSP обеспечивает большую расширяемость, позволяя пользователям добавлять большее количество клиентов, и лучшее обслуживание сетей. Система является надстройкой над протоколом IPX. Планируется полностью интегрировать систему в следующую реализацию NetWare 4.1 и продукты NetWare MultiProtocol Router.

Образовательная программа «The New Novell»

Анонсируя «новую эру» в обучении информационным технологиям, «Novell Education» представила в 1994 году детали относительно новой глобальной организации и некоторые программы, связанные с интеграцией традиционного обучения «Novell» с «WordPerfect Global Education Development Group». Для обеспечения пользователей требуемыми им повсеместно знаниями о сетях и приложениях для сетей, «Novell Education» представила две категории обучения, деловую и профессиональную, как расширение существующих программ и новой философии «открытого рынка» в обучении конечных пользователей. ■

Все упомянутые в блоке новостей торговые марки, зарегистрированные торговые марки, товарные знаки и зарегистрированные товарные знаки принадлежат их уважаемым владельцам.

С 10 по 14 мая 1995 г. в Санкт-Петербурге состоится

X Международная конференция по М-технологиям,

посвященная промышленным информационным системам.

Телефон: (812) 218-71-22.
Факс: (812) 243-20-96.

Председатель оргкомитета — Лебедев Виктор Николаевич,

секретарь оргкомитета — Дружинин Андрей Сергеевич.

VIMCOM

Надежный партнер в бурном море компьютерного бизнеса

VIMCOM

ПРОДАЖА

Компьютеры любой конфигурации (586\ISA)
Периферийные устройства Hewlett Packard
Копировальные аппараты Canon
Телефоны, телефонные станции Panasonic
Калькуляторы
Видеотехника DAEWOO, SHARP
Тел.: (95) 306-6210, 176-1249, 181-9872
Факс: (095) 176-7998, 306-6310

СЕРВИС

Коаксиальные и оптоволоконные сети "под ключ"
Mini АТС "Panasonic"
Ремонт любой оргтехники
Тел.: (095) 368-4201

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Автоматизированная система управления банком АСУБ "VIMCOMBANK"

Банковская почтовая машина "АСТРА"
Тел.: (095) 368-4121

Система управления предприятием "ИНФА-ЭКСПРЕС"
Тел.: (095) 176-0560

МонитОр

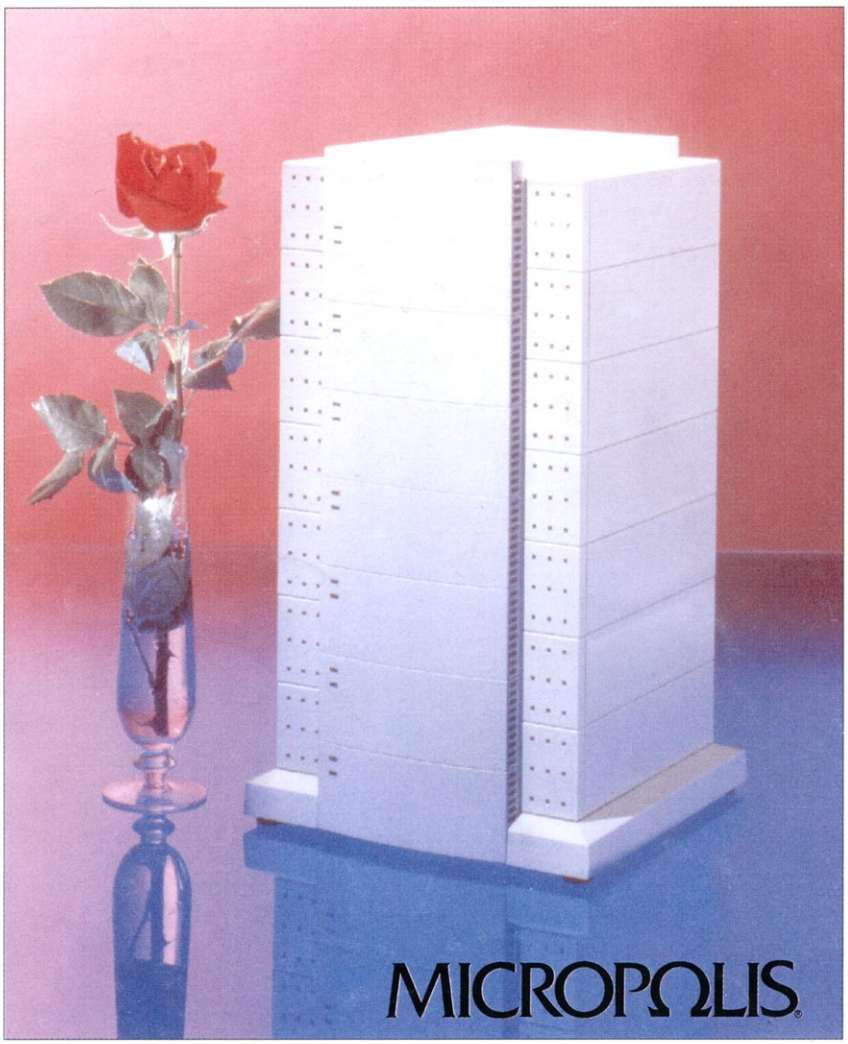
Tested and Approved

Суперъёмкость и супернадёжность

Модульные дисковые массивы Raidion LT и LS - уникальные системы для хранения сверхбольших объемов данных в сетях, работающих под управлением Novell NetWare 3.12/4.0, LAN Server или IBM OS/2. Все модели поддерживают технологию защиты данных RAID 5.

Максимальная емкость устройств Raidion - 93 Гбайта, при этом среднее время поиска не превышает 16 мсек. Скорость передачи данных через SCSI интефейс - от 5 до 16 Мбайт/сек в зависимости от модели.

Техническое обслуживание и замена отдельных модулей производиться при включенном устройстве.



АО Merisel: Тел. (095) 276-9008 (5 линий); Факс (095) 276-4714, 274-0097
Представительство в Санкт-Петербурге: Тел. (812) 294-6012, 298-8429; Факс. (812) 355-8965
Представительство в Казани: Тел. (8432) 533-297; Факс (8432) 533-297
Представительство в Магнитогорске: Тел. (3511) 370-523; Факс (3511) 377-541

MERISEL
World Class Distribution